

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

**RSLES: AN ARCHITECTURAL
IMPLEMENTATION OF A DECISION SUPPORT SYSTEM
FOR OPTIMAL RECRUIT STATION LOCATION**

by

Dale E. Houck and Mark V. Shigley

June 1999

Thesis Advisor:
Second Reader:

Daniel R. Dolk
Kevin R. Gue

Approved for Public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE June 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE RSLES: AN ARCHITECTURAL IMPLEMENTATION OF A DECISION SUPPORT SYSTEM FOR OPTIMAL RECRUIT STATION LOCATION			5. FUNDING NUMBERS	
6. AUTHOR(S) Dale E. Houck and Mark V. Shigley				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for Public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) <p>This thesis describes a component-based methodology for developing a decision support system (DSS) for optimal location of military recruiting stations in regional recruiting markets. The DSS is designed to ensure that stations are selected that minimize cost for a given level of production. The interface allows users to perform "what if" analysis to determine if there are better locations to meet desired objectives. The Recruit Station Location Evaluation System (RSLES) integrates a user interface, a database, a GAMS optimizer model and a geographic information system (GIS) mapping engine to provide a flexible environment that leverages operational recruiting, market analysis, and demographic information for decision-making.</p>				
14. SUBJECT TERMS Decision Support Systems, Geographic Information Systems, Application Development, Recruiting, Site Location			15. NUMBER OF PAGES 187	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18 298-102

Approved for Public release; distribution is unlimited.

**RSLES: AN ARCHITECTURAL IMPLEMENTATION OF A DECISION
SUPPORT SYSTEM FOR OPTIMAL RECRUIT STATION LOCATION**

Dale E. Houck
Major, United States Marine Corps
B.S.B.A., Shippensburg State College, 1983

Mark V. Shigley
Major, United States Marine Corps
B.S., United States Naval Academy, 1985

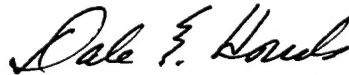
Submitted in partial fulfillment of the
requirements for the degree of

**MASTERS OF SCIENCE IN
INFORMATION TECHNOLOGY MANAGEMENT**

from the

**NAVAL POSTGRADUATE SCHOOL
June 1999**

Authors:



Dale E. Houck



Mark V. Shigley

Approved by:

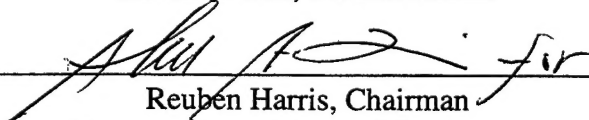


5/21/99

Daniel R. Dolk, Thesis Advisor



Kevin R. Gue, Second Reader



Reuben Harris, Chairman
Department of Systems Management

ABSTRACT

This thesis describes a component-based methodology for developing a decision support system (DSS) for optimal location of military recruiting stations in regional recruiting markets. The DSS is designed to ensure stations are selected that minimize cost for a given level of production. The interface allows users to perform "what if" analysis to determine if there are better locations to meet desired objectives. The Recruit Station Location Evaluation System (RSLES) integrates a user interface, a database, a GAMS optimizer model, and a geographic information system (GIS) mapping to provide a flexible environment that leverages operational recruiting, market analysis, and demographic information for decision-making.

TABLE OF CONTENTS

I. INTRODUCTION.....	1
A. INTRODUCTION.....	1
B. PROBLEM.....	2
C. METHODOLOGY.....	2
D. SCOPE.....	3
E. ORGANIZATION OF THE STUDY.....	3
II. PROBLEM EVALUTION.....	5
A. INTRODUCTION.....	5
B. REQUIREMENTS DEFINITION	6
C. DATA REQUIREMENTS	10
D. SUMMARY	12
III. RSLES SYSTEM ARCHITECTURE	13
A. INTRODUCTION.....	13
B. RSLES SYSTEM ARCHITECTURE	15
C. SUMMARY.....	21
IV. APPLICATION DEVELOPMENT.....	23
A. INTRODUCTION.....	23
B. RAPID APPLICATION DEVELOPEMENT	24
C. SUMMARY	39
V. USE-CASE ANALYSIS	41
A. INTRODUCTION	41
B. JOINT PLANNER CASE.....	42
C. SINGLE SERVICE PLANNER CASE	52
D. SUMMARY.....	58
VI. DATA QUALITY AND FUTURE ENHANCEMENTS.....	61
A. INTRODUCTION.....	61
B. DATA ANOMALIES.....	61
C. FUTURE RSLES DEVELOPMENTS.....	65
D. CONCLUSION.....	68

VII. CONCLUSIONS.....	69
A. INTRODUCTION.....	69
B. CONCLUSIONS.....	69
C. SUMMARY	71
APPENDIX A. RSLES SOURCE DATA FILE DATA DICTIONARY	73
APPENDIX B. PROTOTYPING PHASE MOCKUPS.....	75
APPENDIX C. "MIN COST" SOURCE FILE LISTING	83
APPENDIX D. RSLES SOURCE CODE LISTING.....	87
APPENDIX E. INSTALLATION AND SYSTEM NOTES	163
LIST OF REFERENCES.....	167
INITIAL DISTRIBUTION LIST	169

LIST OF FIGURES

Figure 2-1. Sample of MapInfo Thematic Map	9
Figure 3-1. RSLES Architecture Development	14
Figure 3-2. Components of a Decision Support System	15
Figure 3-3. RSLES System Architecture.....	16
Figure 4-1. Linear vs. RAD Development Methodologies	25
Figure 4-2. RAD Using Interactive Prototyping	27
Figure 4-3. RSLES Macro-Architecture Diagram	33
Figure 5-1. RSLES Main Menu	42
Figure 5-2. Joint User Select Box	43
Figure 5-3. MapInfo Tool Bar	44
Figure 5-4. RSLES Metrics Drop-down Box.....	45
Figure 5-5. Joint User Thematic Map	46
Figure 5-6. Joint planner Optimization Screen	47
Figure 5-7. Joint User Production Dialogue Box.....	48
Figure 5-8. GAMS Interface for Successful Optimization.....	50
Figure 5-9. RSLES Joint Service Optimization View.....	51
Figure 5-10. Main Menu for Army Planner	53
Figure 5-11. Select Screen for an Army Company.....	54
Figure 5-12. Thematic Map of an Army Recruiting Company	55
Figure 5-13. Army Optimization Model Setup	56
Figure 5-14. Production Goal Dialogue Box.....	57
Figure 5-15. Geographical Optimizer Output for Company 1A4	58
Figure 6-1. Mapping of Flawed Data	62
Figure 6-2. Chicago Metropolitan Area with missing zip codes.....	64
Figure 6-3. Modified Chicago Metropolitan Area	64

LIST OF TABLES

Table 2-1. Primary and Secondary Requirements	7
Table 4-1. Macro/Micro Architecture Chart	33
Table 5-1. Military Recruiting Organizational Hierarchies.....	41

LIST OF ACRONYMS

ADO	ActiveX Data Objects
API	Application Program Interface
ARIES	Army Reserve Installation Evaluation System
ATAS	Automated Territory Alignment System
COTS	Commercial Off-The-Shelf
CNRC	Commander, Navy Recruiting Command
DAO	Data Access Objects
DBMS	Database Management System
DMDC	Defense Manpower Data Center
DoD	Department of Defense
DSS	Decision Support System
GAMS	Generalized Algebraic Modeling System
GIS	Geographical Information System
GUI	Graphical User Interface
LAN	Local Area Network
MEPCOM	Military Entrance Processing Command
MSA	Metropolitan Statistical Area
NPS	Naval Postgraduate School
ODBC	Open Database Connectivity
OLE	Object Linking and Embedding
OSD(AP)	Office of the Secretary of Defense (Accession Policy)
RDBMS	Relational Database Management System
RDO	Remote Data Objects
RS	Recruiting Station
RSID	Recruiting Station Identifier
RSLES	Recruit Station Location Evaluation System
SQL	Structured Query Language
USAREC	United States Army Recruiting Command
VB	Visual Basic
VBA	Visual Basic for Applications

ACKNOWLEDGEMENT

First and foremost we would like to thank our families for supporting us in pursuit of our educational goals. Thank you all so much for your faithful support, love, and understanding.

Special thanks to LT Mitchell Kerman, NPS, and Mr. Kay Rigg, DMDC for their assistance in providing programming advice and debugging expertise in Visual Basic. Their insights and guidance were very useful in developing the final product of this thesis.

The development of the RSLES prototype application may not have been possible without the generosity of Professor Hemant Bhargava. He graciously allowed the development team full use of his lab and its equipment which was essential to the success of the overall project.

I. INTRODUCTION

"Make it run; make it right; make it fast; make it small."

Kent Beck, 1995

A. INTRODUCTION

Recruiting has become a critical activity in the maintenance of today's military forces. The problem of recruiting sufficient numbers of young men and women to meet annual accession requirements has become one of the Services' most difficult missions to accomplish. This problem is only exacerbated by a robust economy which makes it more difficult to attract and retain qualified persons. During fiscal year 1998, the Navy missed its recruiting goals by 6,892 and the Army by 776, while the Air Force and the Marine Corps achieved their recruiting goals (Maze, 1999).

One way to address the problem is to open more recruiting stations. This might not be practical from a cost standpoint, and the services might not have the additional recruiting personnel available to staff these facilities. The Department of Defense (DoD) has considered many initiatives to improve productivity as well as to make the recruiting process more efficient. One of the ingredients missing in DoD is a mathematical model to locate military recruiting stations optimally. In 1997 the Office of the Secretary of Defense (OSD) commissioned development of an optimization model for locating recruiting stations for a specified region, and for allocating the number of recruiters per branch of service assigned to that station. This more rigorous approach can potentially improve productivity by assigning stations to locations where recruiters have greater access to the 17 to 21 year old population group, the primary age group targeted by recruiters.

Relocation decisions are inherently complicated and have been the subject of extensive study in operations research for many years. Factors such as area cost, number of high school graduates, population of 17 to 21 years olds, and number of high schools within a certain distance of the proposed recruiting station are just a few of the

considerations when making relocation decisions. The situation becomes even more complicated, however, by factors such as joint recruiting efforts where the effects of moving a Navy station, for example, may have an impact on adjoining Army stations. Questions that arise in this context include: How much effect? How will long-term leases now be affected? What facilities are available? Are facilities located in such a way as to allow the service or services to meet target production goals at lowest cost? Because the issues are complex and interrelated, analytical and computer support can significantly benefit decision making.

B. PROBLEM

The problem addressed in this thesis is to design a spatial decision support system (DSS) in a compressed timeline with limited resources that aids DoD decision-makers in answering the following questions:

- Given a target production level, what is the least cost configuration of stations and recruiters required for a given geographic area?
- What are the minimum number of service recruiting personnel required to meet target production levels?

C. METHODOLOGY

The Recruit Station Location Evaluation System (RSLES) DSS must incorporate several components to be effective, including a user interface, an underlying database, a mapping engine, and an optimization model. Once these components are selected, a methodology must be employed that effectively allows design and development of the RSLES application. The primary objective of this thesis is to describe the development procedures used in Rapid Application Development (RAD) software development methodology as compared to the more traditional development paradigms such as the waterfall or spiral method. The research question that will be addressed is:

"What conditions tend to lead to successful implementation of a RAD-based software development methodology?"

D. SCOPE

This study will focus on the DSS developed for OSD Accessions Policy (AP) to support the recruit station location decision problem. Specifically, it will address the Rapid Application Development software development methodology utilized to develop the RSLES graphical user interface (GUI). The RSLES DSS only considers Army and Navy recruiting commands. The service options available are at the company/zone level and the battalion/district level. At the joint level, the RSLES DSS considers 11 metropolitan areas. The optimization model of the DSS was developed concurrently in another Naval Postgraduate School (NPS) thesis (Martin, 1999).

The long-term objective is to incorporate Air Force and Marine Corps data as well as the top 125 metropolitan areas within the United States. Data was taken from the service recruiting commands and incorporated into a database. Time did not permit development of a data administrator or data warehouse. Hence, the underlying database must be updated manually.

E. ORGANIZATION OF THE STUDY

The remainder of this study is organized as described below. Chapter II discusses the problem evaluation, including a problem description, project requirements and database development. Chapter III describes the architecture of the RSLES prototype project and how it was selected. Chapter IV discusses the software development process and how these application development techniques were implemented in the RSLES DSS application. Chapter V is a use-case analysis that walks the reader through a typical user session to determine optimal recruit station locations based on constraints supplied by the user. Chapter VI discusses the lessons learned in developing the RSLES application and using a rapid application development approach to creating an application within a short time period. It presents problems encountered in the rapid application development

approach. Problems with the availability and quality of the database information are also discussed. Chapter VII offers conclusions and recommends topics for further research and development.

II. PROBLEM EVALUATION

A. INTRODUCTION

The National Defense Authorization Act of 1996 included a request for the Department of Defense to conduct a "Joint-Service study for determining the location of recruiting stations and the number of military personnel required to operate such stations." This study resulted in the initiation of research into the development of a forecasting model to "measure recruiting efficiency by predicting cost and production impacts of management decisions to close, open and relocate recruiting facilities." (OASD-FMP, 1996)

As a part of this research effort, the Naval Postgraduate School was tasked by the Office of the Secretary of Defense (OSD) to initiate the Recruiting Station Location Project. The goal of the project is to "develop methods and models to help determine the optimal number and location of recruiting stations." (Mehay, 1998)

The OSD Recruiting Station Location Project is composed of four constituent parts:

- An econometric model for predicting productivity,
- Cost models for measuring recruiter and station costs,
- Optimization models for determining best station locations, and
- A decision support system (DSS) to integrate the models and their associated data.

The econometric model predicts the effects of recruit station location decisions on the number of contracts produced. The cost model predicts how station costs vary with location. The optimization models utilize the functions from the first two models to predict optimal recruit station locations. The optimization models come in two flavors; maximize production for a given cost, and minimize cost for a given level of production. The Decision Support System uses the minimum cost model, and provides a graphical user interface (GUI) that enables users to interact with the models and gauge the effects of alternative decisions. RSLES was developed to provide the functionality of the DSS portion of the OSD Recruiting Station Location Project.

B. REQUIREMENTS DEFINITION

The first step in the classic software development cycle is the definition of requirements. This is usually the most difficult step to accomplish successfully. It is also the most important step. Poor definition of requirements generally leads to costly and time consuming changes to the design of the software as software development progresses. This precept is taught in all software design or software engineering courses, and was adopted as a first principle by the RSLES development team. For this reason, a RAD approach was selected. The concepts and reasoning behind RAD will be discussed at length in Chapter IV.

1. Identification of Stakeholders

To best determine the requirements for RSLES, the development team had to identify the stakeholders who would be affected by the system, and interview them to find out what features and capabilities they desired. Geographical dispersion, budget constraints, time limitations and unavailability of key personnel made detailed interviews infeasible. Instead, we met with the members of the OSD Recruiting Station Location Project Team from the Naval Postgraduate School and with individuals from the Lewin Group, a firm sub-contracted to conduct regression analysis to determine recruit station cost factors. At this meeting the major stakeholders were identified as the Office of the Secretary of Defense (OSD) for Accessions Policy (AP), the Office of the Assistant Secretary of Defense (Force Management Policy) (OASD(FMP)), the Joint Recruiting Facilities Committee (JRFC), and the Recruiting Commands of the individual services.

2. Requirements Analysis

At the September 1998 meeting with the Recruiting Station Location Project Team, we derived primary and secondary requirements. Primary requirements are those which must be incorporated in the delivered product. Secondary requirements are those that would be nice to include in RSLES, time permitting, after the primary requirements are installed. Table 2-1 identifies the primary and secondary requirements for RSLES.

PRIMARY REQUIREMENTS	
Display Current "as is" Army and Navy information	
Use MapInfo for the mapping engine	
Solicit user for input and generate Generalized Algebraic Modeling System (GAMS) input files	
Given a GAMS optimization model output file, graphically display its results on a map	
SECONDARY REQUIREMENTS	
Provide ability to run the optimization model directly from the GUI	
Automatically extract optimization model output files and graphically display the solution	
Provide a thematic mapping capability to assist the user in making decisions	
Incorporate Air Force and Marine Corps data into the RSLES database and GUI	

Table 2-1. Primary and Secondary Requirements

The first primary requirement is that RSLES must display current "as is" Army and Navy information. This is to be accomplished by accessing the underlying database and graphically displaying the current location of Army, Navy and Joint (both Army and Navy) recruiting stations for the selected unit or metropolitan area. At the time the RSLES project was initiated, the Recruiting Station Location Project team had been in contact only with the U.S. Army Recruiting Command (USAREC) and Commander, Naval Recruiting Command (CNRC). Both commands were able to provide zip code level data regarding production, station location, and recruiters assigned. In an effort to limit the scope of the RSLES project to an attainable goal for the time allotted, the development team opted to implement RSLES using Army and Navy information only.

The second primary requirement is to use the MapInfo mapping engine. This requirement was generated by OSD because MapInfo is already being used by the Joint Recruiting Facility Committee (JRFC) as well as most of the services' recruiting commands. It is an extremely useful mapping application that enables database tables with geographic information, such as latitude and longitude, universal transverse mercator (UTM) coordinates, zip codes, and even street addresses to be geographically coded and projected on a map. It also has many powerful features that enable data

associated with the geographic information to be displayed spatially in ways that enable the user to rapidly identify the current situation and trends.

The third primary requirement is to solicit input from the user and generate GAMS input files. GAMS is a mathematical modeling system that uses a high-level language for compact representation of complex models. Like MapInfo, the use of GAMS was specified by OSD. Unlike MapInfo, GAMS is DOS-based, and was not designed to be accessed through a Windows-based GUI. The intent for this requirement was that at a minimum, RSLES would query the user via input screens, and generate the text files necessary for GAMS to run an optimization solution off-line.

The fourth primary requirement is for RSLES to read GAMS output text files and graphically display the optimization model output. Again, this requirement was established as a minimum in light of the complexities of running DOS-based applications under a Windows-based GUI.

These four requirements describe what the minimum deliverable should look like. This scope was made intentionally narrow to allow us to focus our efforts and provide the deliverable in the amount of time available. The secondary requirements were derived as additional features to incorporate into RSLES if the primary requirements were met and time permitted their development.

The first two secondary requirements are intended for RSLES to automatically make the interface to GAMS transparent to the user. They require automated feeding of the input files into GAMS as well as automated retrieval of the GAMS output file. These requirements will free the user from having to manipulate files and manually run GAMS off-line, detecting when GAMS has finished running, and graphically displaying its output. If both of these requirements are met, the user would not be required to be familiar with GAMS to run the optimization model.

The third secondary requirement is for RSLES to provide a thematic mapping capability. This feature would create color-coded maps based on a metric that the user identifies from a selection. The resultant map would clearly identify trends and situations that may not be easily discerned from tabular data. A thematic layer allows for information to be displayed by special symbols or color-coding of regions. Figure 2-1 provides a sample of a MapInfo thematic map layer based on 1990 State populations.

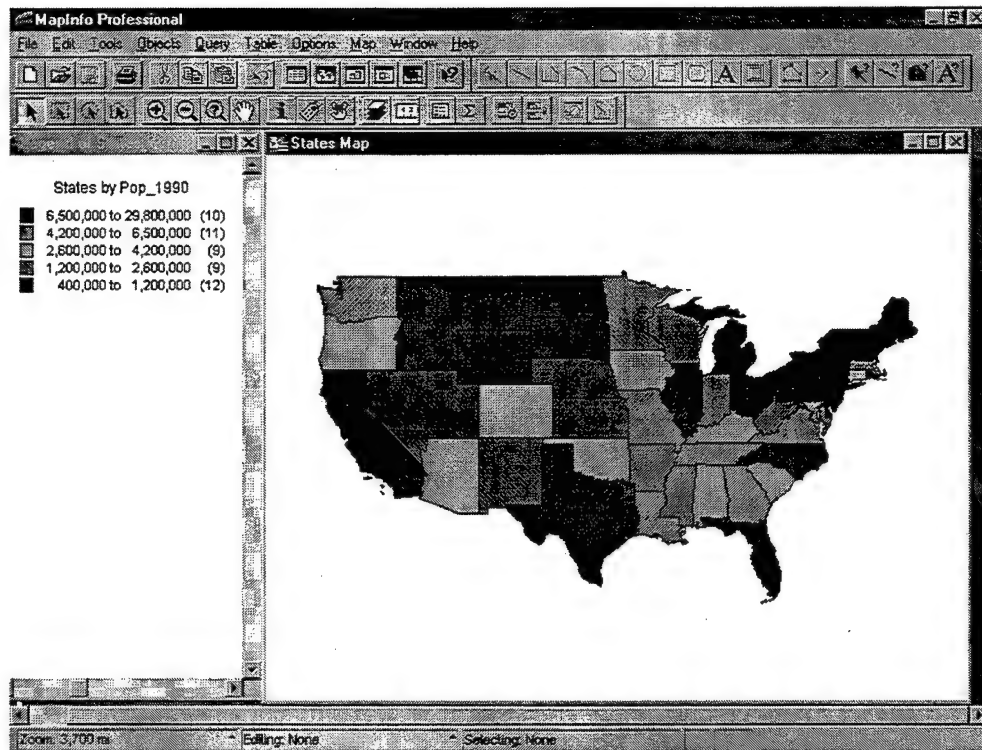


Figure 2-1. Sample of MapInfo Thematic Map

The last secondary requirement is to incorporate U.S. Air Force and Marine Corps recruiting data into RSLES. This is an important requirement; however, data for these services were not available during the requirements definition phase or even the design phase. This is the only secondary requirement that was not met and can serve as a follow-on research project.

Shortly after project initiation, we traveled to USAREC at Ft. Knox, KY to obtain user feedback from a mock-up of the RSLES interface. This "Version 0" prototype was presented to the marketing section in order to ensure that we were designing a product that would meet USAREC's needs and to elicit ideas regarding any additional features they would like to have implemented. They expressed some disappointment that RSLES was not yet functional, but were satisfied with the requirements as enumerated above. One month later, the RSLES development team met similarly with representatives from CNRC who were also satisfied with the requirements as they were stated.

C. DATA REQUIREMENTS

The validity and quality of the source data directly affects the quality of the eventual utility of the RSLES DSS application. An understanding of the data problems involved in the RSLES project did not emerge until the application was in full development.

At the time of the initiation of the RSLES project, the Recruiting Station Location Project team had been using a vast set of data collected by the Lewin Group from numerous sources including the Military Entrance Processing Command (MEPCOM), USAREC, CNRC, Woods and Poole, MapInfo, the Census Bureau, and the Bureau of Labor Statistics. This data was amassed to run regression analyses to determine the beta values for the optimization model. This enormous data set consisted of eight quarters of zip code level data containing Army and Navy Recruiting Station territory alignments, and locations, population data, and recruiting accessions information. This data set was stored as a very large flat file consisting of nearly fifty variables and approximately 400,000 records. The file was more than one Gigabyte in size, and was somewhat difficult to transport from one computing platform to another. In order to access the file, we reduced its scope to the most recent quarter and approximately 25 variables. We decided it was unnecessary to maintain this amount of historical data since each service already maintains this data. Further, since the purpose of RSLES is to determine potential recruiting station locations, historical data beyond a year old does not have significant value. Chapter VI details problems encountered with the original dataset. These problems were severe enough to cause us to discard the original data and collect an entirely new set.

1. Data Collection

The collection of data was an iterative process of gathering from one source at a time, organizing it, then joining it with the data from the next source. The first set of data was provided by USAREC. Their Automatic Territory Assignment System (ATAS) includes a wealth of data that not only provides zip code level data for Army recruiting stations, but also contains accessions information, by zip code, for all four services. This

accessions data was provided to USAREC by the Military Entrance Processing Command (MEPCOM). ATAS also had Woods and Poole zip code population data, as well as high school and college data.

The bulk of the information provided by USAREC and used in RSLES came from two tables. The first provided information on the zip codes themselves, including the preceding information as well as recruiting station alignment information. The second table provided information about each of the Army's recruiting stations, such as its location, its identification number, its company, battalion and brigade, and the number of recruiters assigned. In order for the data to be used by RSLES, these tables had to be joined into a single table using dummy variables to indicate which zip codes had recruiting stations in them.

The second source of data was CNRC. They also provided their data in two files; a Navy Station file and a Navy zip code file. This data was joined into the same table with the ATAS data. The Lewin Group provided the third source of information. They furnished the zip code level econometric data required by the optimization model. This information included unemployment rates, per capita income, regional designators, and dummy variables to designate whether a zip code is urban, suburban or rural. Finally, Census Bureau information was obtained to designate zip codes by their metropolitan area. This designation is required for joint recruit station optimization planning. The variables used and their definitions are contained in Appendix A.

2. Data Quality Analysis

When collecting and implementing the data, we made three assumptions regarding the relationships between zip codes and recruiting stations. These assumptions are:

- Every zip code will be assigned to a recruiting station,
- Every recruiting station will have zip codes and recruiters assigned to it, and
- Every recruiting station will be in a zip code that is assigned to it.

The quality of the data can only be qualitatively estimated by the presence or absence of anomalies to the assumptions listed above. In the case of the data provided by USAREC, fewer than 20 out of over 1500 stations violated any of these assumptions. The data

provided by CNRC contain nearly 200 out of 1200 stations that violate one of the three assumptions. However, according to CNRC, these anomalies are not erroneous, they simply reflect stations that are temporarily unmanned, or are not located within their own territory. The effect these unusual stations (and RSLES' method of modeling them) have on the accuracy of output from the optimizer model is difficult to measure. However, RSLES is a DSS, not a decision-maker. For example, RSLES cannot accurately measure the impact of a recruiting station with no recruiters assigned. Those making recruiting station location decisions must consider the assumptions that RSLES operates under and inject their own experience into their deliberations.

E. SUMMARY

The RSLES DSS project encountered two challenges common to prototype application development. Although the intended users of the system are known, they were never truly involved in determining the requirements, due to time constraints. It was a matter of time and funding available to the developers, and competing requirements for the users.

The second obstacle concerned the acquisition of data. Database tables as large as those utilized in the RSLES project frequently have to be examined through different views before problems can be identified. The RSLES development team underestimated the amount of time that would be necessary to collect, assemble and inspect clean data. The additional time devoted to data preparation limited the number of "nice to have" features that were identified in addition to the system requirements.

With the project requirements defined and the quality of the data now known, the next step was to review the architectural components and consider the best method to design the system. The next chapter describes how the RSLES system architecture was defined and the considerations that went into its design.

III. RSLES: SYSTEM ARCHITECTURE

A. INTRODUCTION

This chapter describes the architecture of the RSLES DSS project developed at the Naval Postgraduate School (NPS) for the Director of Accession Policy in the Office of the Secretary of Defense (Force Management Policy). A software architecture defines the common structure of a system by specifying the components that comprise the system, the relations and interactions between the components, and the rationale for the design decisions embodied in the structure (Luqi, 1995).

The aim of software development is a concrete representation of the software that executes within a system to provide a desired function (Waugh, 1995). We consider two levels of architectural abstraction: macro, which is an overall description of the entire system, and micro, where we describe the system with its components.

While there is no single development method or architecture that can guarantee a smooth or simple development process, judicious selection of a development methodology and supporting tools, coupled with a thorough identification of requirements can go a long way towards increasing the chance of project success. The RSLES Development Process is depicted below in Figure 3.1. The NPS development team sought to minimize application development time, which argued strongly for the use of COTS applications and Service historical databases as the primary components of its infrastructure.

Additionally, the team sought to avoid developing a strictly requirements-driven application. It was considered more important to focus on the pieces of functionality with the highest business value, and deliver that functionality rapidly. An example is the 'Display Optimizer Output' module. Since this was the most important in terms of development priority, we focused initially on its development. Changes in requirements are often the reason for delays in application development. This problem is exacerbated during long linear development processes, where changes in functionality requirements or

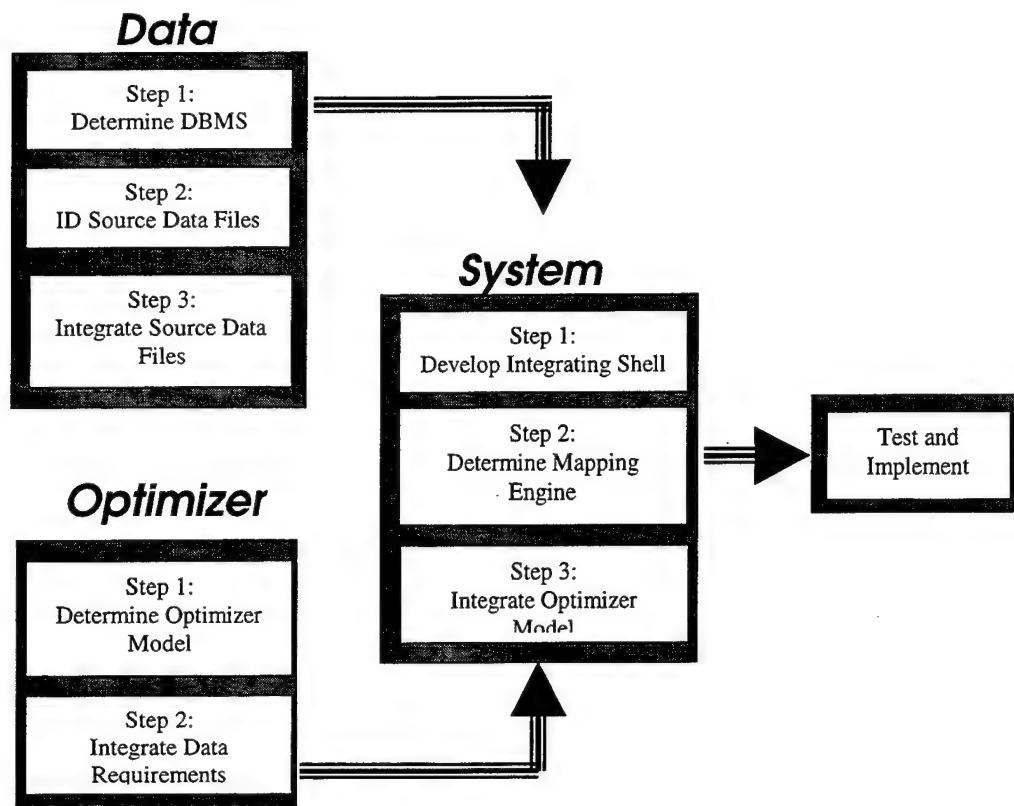


Figure 3-1. RSLES Architecture Development

project scope can cause many months to be lost and significant expense to be incurred for redesign and redevelopment. This is especially true when substantial time has been invested in planning, design, development, and testing.

The first priority was to develop a Data Model from the disparate data sources containing service-recruiting information. Once the Data Model was specified, an automated decision application could then be developed to integrate the optimizer model's information needs with the available information in the data model. The Graphical User Interface (GUI) accepts the required inputs for the problem from the user and puts the information in the required GAMS format before calling the optimizer model.

B. RSLES SYSTEM ARCHITECTURE.

1. DSS Architecture

RSLES is a unique project in that it was developed to fulfill a function that was previously not available. Its objective is to assist the decision-maker involved in recruit station location decisions. DSSs are computer-based information systems that provide interactive information support to decision-makers during the decision-making process. Decision support systems use (1) analytical models, (2) specialized databases, (3) a decision-maker's own insights and judgments, and (4) an interactive, computer-based modeling process to support semi-structured and unstructured decision-making by individual managers (O'Brien, 1993). They are designed to be ad-hoc, quick response systems that are initiated and controlled by end-users. The components of a decision support system are shown in Figure 3-2.

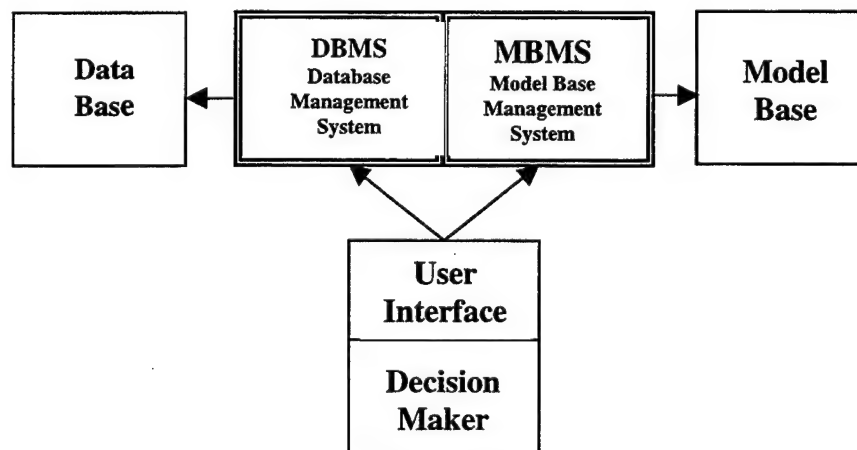


Figure 3-2. Components of a Decision Support System

2. Background

It became clear early in the project that budget and time limitations would not allow the development of an application that would carry out all functions of this project

independently. This led to the integration of several commercial of-the-shelf (COTS) products to conduct the optimization and assist in the GIS portion of the project. The RSLES application architecture consists of four components: an integrating shell, a mapping engine, an optimizer model, and a database management system (Figure 3-3).

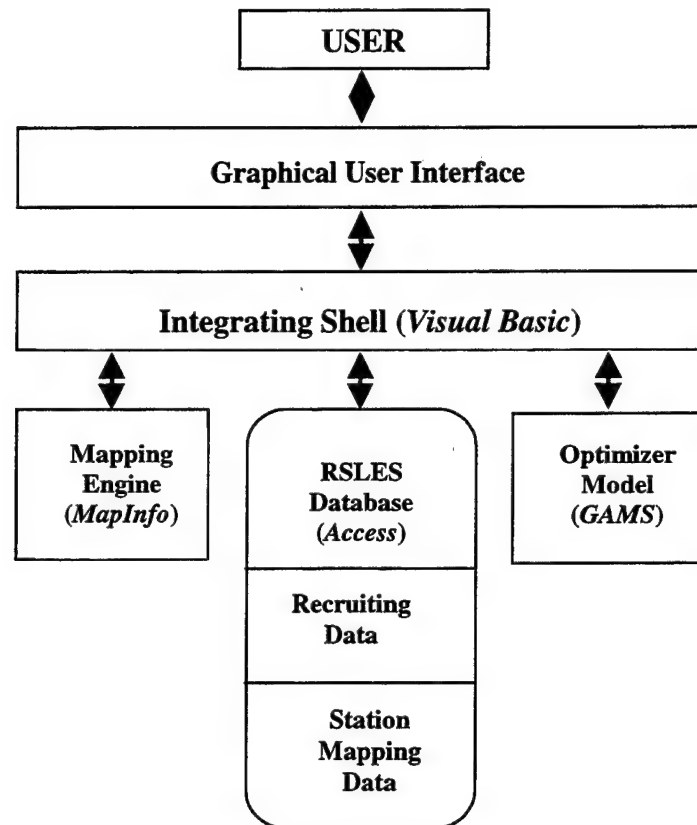


Figure 3-3. RSLES System Architecture

At the heart of the RSLES architecture is an optimizer model that was developed to minimize recruiting costs subject to production requirements. The optimizer model chooses from among a list of candidate zip codes within a defined geographical area and identifies which ones should contain recruiting stations as well as the optimal number of recruiters associated with each station. This information is then mapped to operational source databases and graphically displayed. The database contains the source data necessary for the mapping process and for the optimizer model to run.

3. Integrating Shell

The application shell that integrates the COTS products and operates the GUI is original code written in Visual Basic. Visual Basic is an event-driven programming language that has many object-oriented features and is excellent as a RAD tool for developing integrated applications. In the event-driven model, programs are no longer procedural, and they do not follow a sequential logic (Bradley, 1998). Rather, the user determines the sequence of execution dynamically. An additional benefit of using Visual Basic is that several primary users, such as USAREC, already have information system support personnel with Visual Basic programming experience. This provides an opportunity for future prototype maintenance and improvements to be completed in house by those experienced programmers.

Another requirement for the prototype was that the final application should relieve the user from the burden of understanding the individual COTS applications and protocols involved in the transfer of information. Because of the limited and structured nature of the decision process used to determine Recruit Station location, automation of most of the tasks was necessary. This included passing data to the optimizer, geo-coding, and creating thematic maps for the visual display of metrics associated with individual recruit stations. The only exception to this requirement in RSLES is that the user must be somewhat familiar with the basic toolbar functions of MapInfo such as select, zoom-in, zoom-out, and layer control.

Required inputs from the user are minimal and include proposed zip codes where a station should be opened, where a station should be closed, or 'candidate' zip codes where the user allows the optimizer model to consider specified zip codes for closing or opening. Appendix B (User Interface Final Layout) shows the RSLES User Interface screens used to capture the input parameters.

The GUI accepts these inputs and then uses them in an SQL query to identify the characteristics associated with each zip code (i.e. total high schools, population density, etc.) where a station is currently located. This information is then combined with the

domain's (i.e. Battalion, District, or Metropolitan area) characteristics and is then used to create GAMS input files in Visual Basic (GAMS Source File listing is provided in Appendix C). The input files are then passed into the optimizer using a shell. These tasks are carried out through Object Linking and Embedding (OLE) automation with the mapping engine. Others are carried out using the database engine in Visual Basic. Automation (formerly OLE Automation) is an industry standard used by applications that allows objects to be shared by many applications (Figure 3-4).

4. Mapping Engine

MapInfo was chosen as the mapping engine for several reasons. MapInfo satisfied all the known and anticipated functional requirements, is well supported and documented, and minimizes the need for additional training, since it is already in use at each Service's Recruiting Commands. Additionally, version 5.0 of MapInfo allows Visual Basic to use MapInfo tables as bound data control objects. It also allows for direct export of data tables to Microsoft Access and other database applications.

MapInfo is a commercial mapping package that serves as a graphical input tool and a mechanism for the spatial definition and processing of data. It converts positions to distances, makes proximity determinations, and classifies objects by geographical region. The integrating shell uses an automation object to pass data to and from MapInfo and to execute queries in MapInfo. The ability of MapInfo to localize data from huge databases provides a significant performance gain when certain queries are implemented.

5. Optimizer Model

The optimizer model for the RSLES application was developed using GAMS. GAMS was specified by OSD for implementation because of its powerful solver capabilities and its ability to provide a flexible decision analysis environment. The optimizer model has the primary objective of minimizing DoD cost subject to a target production goal. Another optimization model was developed (Martin, 1999) that has the primary objective of maximizing the number of contracts from a region subject to a

budget constraint (Max-Production model). Due to time constraints, RSLES only incorporates the default model, which is the Minimum-Cost model (referred to as "Min-Cost" model).

GAMS presents several obstacles to communicating with other applications, primarily because it is a DOS-based application. Fortunately, Visual Basic provides a 'shell' function, which allows the ability to interface with DOS-based applications. The RSLES application must pass control to GAMS in order to run the optimizer. This requires that text files be created to include the data positioned in the format required by the Min-Cost model. In total, seven files are required as input to the Min-Cost Model (Appendix C). The files include data regarding the domain of zip codes, attributes of each zip code in the domain, units selected, zip codes where stations should be closed, opened, or zip codes where a station can be "considered" for opening or closing. Some of the files contain attribute data about the domain being optimized (i.e. battalion, district, company, or zone) while others contain attribute data for only the zip codes under consideration. While cumbersome, this proved to be the only method to pass the required data to GAMS.

Another limitation is the maximum size of the problem the Min-Cost model is capable of considering. Although it would be beneficial for planners to be able to optimize RS locations for district/battalion sized areas, problems of this size are computationally intractable. The current Min-Cost model solves problems at the company/zone level (for single-service scenarios) and the metropolitan area level (for joint scenarios).

Fortunately, displaying Min-Cost model output was less of a challenge. The optimizer model output is in a text-based format, allowing the RSLES shell to pass the file to MapInfo for evaluation and geographic display. This required several steps, including geo-coding the necessary tables. While this functionality is built into the MapInfo application, we preferred to automate the process for the user, thereby saving a significant amount of time anytime the process was necessary. However, from a coding standpoint, this process proved extremely difficult because the MapX program was not

available for use. Ultimately, we were able to automate the geo-coding process, which proved to be the most difficult module to develop during the coding process.

6. Database Management System

The final component of the architecture is the database management system (DBMS) for which we selected Microsoft Access 97. There were many reasons for its selection, however the overriding factor is its ready availability to the end users who will be using RSLES. When considering the necessary tools for managing databases on Windows-based PCs, the benefits of Access 97 best fit the requirements. For example, it is a true 32-bit DBMS with a multi-threaded, 32-bit database engine (the Jet 3.5 engine). The Jet Engine is a well-designed relational database engine and is shared across Microsoft products, most notably Visual Basic. It is also very extensible, including support for 32-bit OLE custom controls, the Windows 32 API, and VBA add-ins (Litwin, 1996). Also important is the fact that it has an excellent object model for the manipulation of data using Visual Basic code: Data Access Objects (DAO). This, combined with its support for SQL and dynasets, which are a temporary set of data taken from one or more tables in an underlying file, made it the most sensible choice for this project.

Initially, an attempt was made to rely solely upon MapInfo's DBMS to remove one less object from the architecture. While the properties of data bound controls could be set to MapInfo tables, we were unsuccessful in integrating the two applications in other areas where MapInfo-specific code (MapBasic code) was required. Although it may be possible, and perhaps even more efficient, time constraints prevented such an implementation. Additionally, the inconsistency of source databases as well as their numbers and sizes, would have posed considerable implementation challenges for such a design interface.

C. SUMMARY

The RSLES architecture has the major benefit of already having the necessary infrastructure in place at many commands that will be using RSLES. The choice to use a hybrid solution, combining various COTS components, was necessary because no one COTS product exists that satisfies all of the requirements. However, by using several applications that can be nearly seamlessly integrated to provide a total solution, the likelihood of meeting all requirements can be increased. The disadvantages are that broader product knowledge is required during development and the mechanics of package integration often proves difficult, particularly with respect to data transfer between applications. Additionally, open database connectivity, object linking and embedding, dynamic link libraries, and other integration tools necessary for multiple COTS applications add a layer of complexity that can adversely affect performance and reliability. Finally, version maintenance of the COTS products, once the system is deployed, becomes the counterpart of the traditional software maintenance problem. The configuration of new versions as they become available can consume substantial amounts of time as the system evolves.

The DSS architecture of an optimizer model, mapping engine, database management system, and an integrating application system provides the necessary tools for recruit station location decision support. Combining these components provides process functionality that previously did not exist.

With the system architecture defined, the next step is the development of the application. The following chapter will outline the Rapid Application Development (RAD) methodology used to develop RSLES.

THIS PAGE LEFT INTENTIONALLY BLANK

IV. APPLICATION DEVELOPMENT

A. INTRODUCTION

During the 1980's and early 1990's, software process models used one of four major models (code-and-fix, waterfall, evolutionary development, transform) to determine the order of the stages involved in software development and to establish the transition criteria for progressing from one stage to the next (Oman, 1990). Most of these methodologies used similar systematic steps: analysis, design, construction, testing, and maintenance. This was sufficient as long as the user's requirements were well known and the system architecture was well understood. Stage development was generally done in the same order. A by-product was that development generally took years instead of months. Many of the deficiencies of these models were addressed with the introduction of the spiral model (Boehm, 1998), however, it too requires a lengthy development period and is more difficult than the traditional waterfall method to manage from a project management standpoint.

While there are many software development paradigms available for application development, no single method can ensure successful accomplishment of designated objectives guarantee the development team a smooth or simple method of development. A rapid application development (RAD) approach was selected by the project development team because it contains the necessary steps to meet Service requirements, integrate various COTS products using a fourth-generation language (4GL), and meet the constraints imposed by the project.

This chapter begins with an overview of software development, specifically RAD, and documents why it is appropriate for applications must be built in a short timeframe with minimal requirements. This is followed by a discussion of the generic requirements of each of the RAD development phases, and how they relate to the RSLES implementation. The final section is a summary of the RSLES implementation using the RAD heuristic process.

B. RAPID APPLICATION DEVELOPMENT

1. Overview

A major disadvantage of information system development is that it takes too long to implement full production products from untested, basic concepts. RAD is a high-speed waterfall development method using component-based construction. Component-based development, which is the ability to design and build from definable objects without the benefit of inheritance, is a recent and significant advance in software development. The idea of developing applications with reusable code has come to prominence with the advent of object-oriented design. Component tools emphasize assembling applications out of pre-existing objects rather than writing code, while RAD tools focus on ease of use. The RAD approach is now the standard for developing applications quickly, especially when requirements are not well defined and development time is short.

The major reasons for this are:

- The growing use and acceptance of Microsoft's Component Object Model (COM) as the dominant standard for PC-based component computing.
- Components are moving off the desktop and beginning to play an important role in the creation of client/server applications.
- Broad support for designing and developing components and component-based applications with third-party tools and languages. (Chappell, 1997)

Fourth-generation language (4GL) RAD tools provide programmers with a GUI-based environment in which to quickly create applications by choosing from a menu of 'objects' and plugging them in. The benefits of 4GL RAD tools include shorter development life cycles, and greater reliability and stability. 80% of software developers using RAD have chosen Microsoft's Visual Basic (Greenfield, 1997), mostly because the graphic user environment of the 4GL tools enables quick access to application development.

2. Iterative Prototyping

In the traditional, linear or “waterfall” approach to development, users are consulted only during the initial analysis phase with the remaining steps executed in series, with the end users not seeing the application until it is deployed, when changes are usually too difficult to implement (Figure 4.1). RAD methodology executes these steps in parallel, with the working application becoming an aid to analysis and design, by giving users a concrete context for feedback, as opposed to abstract and hard to understand diagrams. This is key to success using RAD; users participate in testing, and development becomes an iterative process of refining successive versions of the application. For these reasons, the NPS development team chose rapid application prototyping for the U/I and the application.

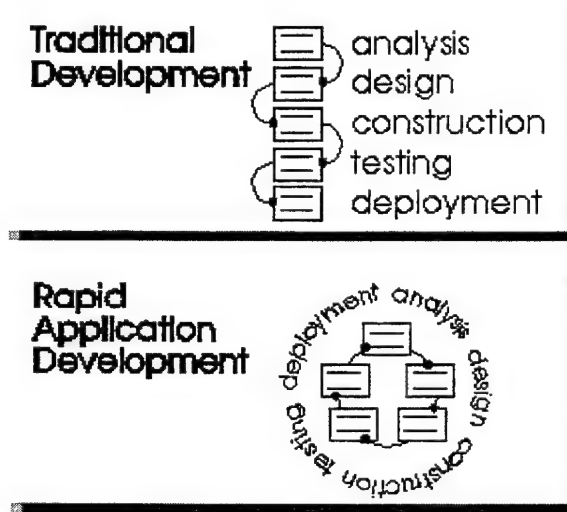


Figure 4-1. Linear vs. RAD Development Methodologies

This type of development process is designed to accommodate pre-existing applications and COTS products. It also employs a systematic development approach that is independent of the type of tool, style, or technique and helps to guide the efforts of the developer and users alike.

RAD uses iterative, evolutionary prototyping to elicit the user’s requirements, refine the prototype, and reconstruct/redesign the product. Ideally, a Joint Application Development (JAD) meeting is held where high-level end-users and designers meet in a

brainstorming session to generate a rough list of initial requirements. Users talk about their requirements and listen to the developer's thoughts and vice versa. After both parties have agreed upon an initial prototype, the process of 'iterating until done' begins. This process generally includes the following: (Maner, 1997)

- Developers build / evolve the prototype based on current requirements.
- Designers review the prototype.
- Customers try out the prototype, change their requirements.
- A Focus Group meeting is held where customers and developers meet to review the product together, refine requirements, and generate change requests.
 - ◆ Developers listen.
 - ◆ Customers talk.
- Requirements and changes requests are "timeboxed".
 - ◆ Changes that cannot be accommodated within existing timeboxes are eliminated.
 - ◆ If necessary to stay "in the box," secondary requirements are dropped.

The sequence of iteration includes re-specify, re-design, and re-evaluate (Figure 4.2). Booch observes that two traits are common to virtually all successful object-oriented systems and noticeably absent from those counted as failures:

- The existence of a strong architectural vision and
- The application of a well-managed iterative and incremental development life cycle (Booch, 1994).

Booch also notes that such a process is iterative in that it involves the successive refinement of an object-oriented architecture whereby the experience and results of each release are applied to the next iteration of analysis and design. It is incremental in that each pass through the analysis/design/construction cycle leads to a refinement of the projects strategic and tactical decisions, ultimately converging upon a solution that meets the end user's real requirements, and yet is reliable and adaptable.

RAPID APPLICATION DEVELOPMENT USING ITERATIVE PROTOTYPING

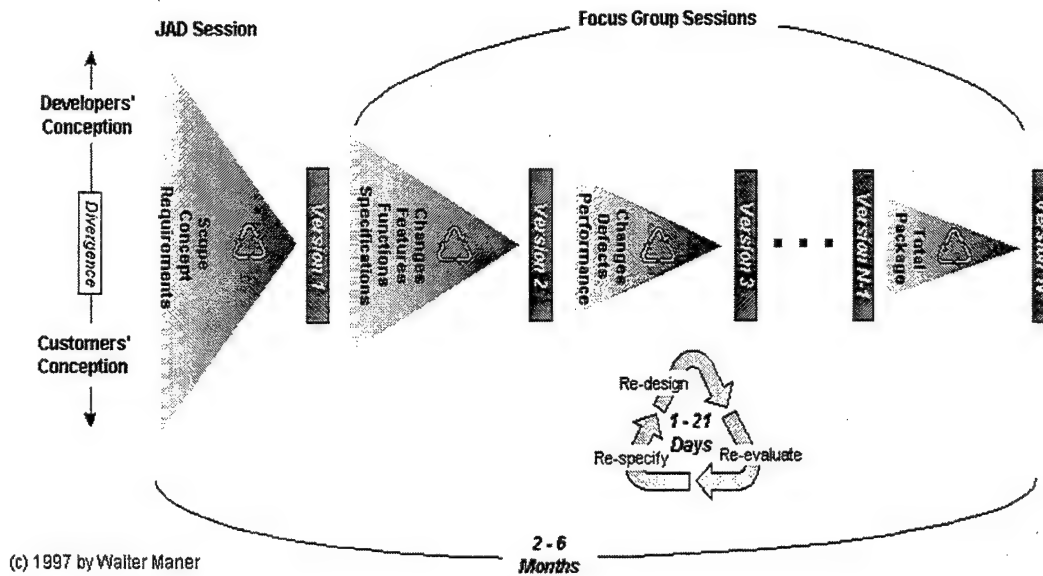


Figure 4-2. RAD Using Interactive Prototyping

In addition to the features outlined above, RAD has some unique principles that aid in the process of development. They include:

- Empowering the RAD team to make decisions.
- Integrating testing into the process.
- Developments kept to short timescales since it is better to deliver a series of small modules than a large complete system.
- Delivering a workable Version 1 system in short order is more important than getting every bit right; Version 2 will be much better if it takes account of user experience over a few months. (MDA Computing, 1996)

RSLES was not developed as an object-oriented project, however the principles and methodology used to develop the architecture are applicable. The following sections describe the development processes used to generate the RSLES architecture.

3. Analysis Phase

The objective of this first phase is to identify the application to be developed, and define it in terms with which the user is familiar with so that it can be used for follow-on software development. This will mirror the traditional phase of analysis:

- Establish the core requirements for the software,
- Develop a model of the system's desired behavior, and
- Create an architecture for the implementation.

Our purpose during this phase is to identify the top-level objectives for the project and to validate its underlying assumptions. A prototype U/I was the primary product of analysis phase.

a. RSLES Application Identification Stage

The project that OSD (AP) was interested in developing was a means by which to display information to support decisions on station locations, specifically optimal station locations as identified by the optimizer models. The display would be the result of a selected unit run through the optimizer model and displayed using MapInfo. The project required a development tool that could integrate the optimizer tool with a mapping engine and DBMS to provide users with the ability to geographically display the results of an optimized recruiting unit, or select a unit for optimization and specify criteria for optimization. The solution was a DSS which would provide a new capability that allows decision-makers to consider the 'optimal' solutions developed by the optimizer models.

b. RSLES Definition Stage

The RSLES system can best be described as a software application that provides the optimal number, location, size and type (single service, joint) of recruiting stations in a given geographic area. It was designed for use at a variety of levels, from

OSD all the way down to the company (zone) level. With this in mind, the following elements were considered during its design:

(1) Top Level Requirements.

- Develop a DSS to provide information to support decisions on station location.
- Operate on standalone IBM PC compatibles running the Microsoft Windows operating system (see System Requirements).
- Implement single user interface that allows the decision-maker to graphically or manually select units for optimization.
- Allow the decision-maker to determine stations to close, open, or fix when optimizing a recruiting area.
- Maximize use of currently available COTS application, specifically MapInfo.

(2) Timeline.

- Develop initial prototype for evaluation within one month.
- Develop a standalone prototype in approximately eight months.

(3) System Requirements.

- *Hardware (standalone)* – IBM PC compatible with minimum 32MB RAM, 150 MHz Pentium processors.
- *Operating Systems* – Microsoft Windows 95, 98, or NT 4.0 or above.
- *COTS* – MapInfo Professional 5.0, GAMS 2.25.089 or above, Microsoft Access 95 or above. Note: Visual Basic is not required.
- *4GL* – Microsoft Visual Basic 6.0 Professional Edition.

c. *RSLES Analysis Prototype*

The requirements for the RSLES project were unclear initially and there was little direction as to how the U/I should appear or function. As a result, we developed a prototype for the RSLES system using our personal experience, consulting prior research on the subject, and considering how recruiting station resource allocation decisions were currently being made. A combination of Visual Basic as the development

decisions were currently being made. A combination of Visual Basic as the development tool, MapInfo as the mapping engine, and Access as the DBMS were used to develop an initial U/I layout (Appendix B).

Prototypes can range in complexity from typical screens to working models that are capable of testing and proving the feasibility of the proposed functionality. We chose the former because it was easier to develop and allowed users to imagine the system in operation. Our objective was to quickly develop a prototype that would stimulate lateral thinking among the users and NPS development team.

Direct interaction with the users of RSLES and the development team was not practical initially; however, we were cognizant of the need to have the system execute in a Windows-based environment and be as automated as possible. Our research indicated that Service-recruiting commands were very familiar with MapInfo and its menu-driven commands, so this allowed us some flexibility when deciding what tools we would make available to the user in the U/I. Our main objective was to display the current station alignment, metrics associated with the current station layout, and optimal station alignment (from the optimizer model). One of the major reasons for selecting Visual Basic as our development tool was its ability to allow event-driven programming which is an integral facet of the project. Users did not desire system output to follow a specified pattern that was the result of a sequential process, but rather to generate output from selection events, regardless of order. Since we were limited to one "master display" window for the map view, allowing the user to perform only one "event" at a time was the agreed upon solution. The initial prototype was presented to users from USARC and CNRC for evaluation. Their recommendations were discussed with the development team and in most cases implemented. The key requirements, views and controls necessary were:

- Allow the user to select a view (Service, Size, and Unit) and graphically display the selected unit.
- "As Is" display. The selected unit is displayed with current station locations. User can select metrics in which to view the selected unit.

- “What If” display. User selects a view, and can optimize the unit and display the results. They have the option to specify zip codes where a station should be opened, a station closed, or zip codes where the optimizer should consider opening or closing a station.
- Button controls to reset the screen view, exit the program, run modules, and search for entered units.
- Toolbar to allow the user to manipulate the map view.

Before proceeding to the next phase, a Systems Requirements Review was conducted with the NPS development team to gain approval of the preliminary requirements and proposed architecture. It was important to ensure that basic requirements were being met while ensuring no additional features were included that could not be delivered in the short timeframe available for development.

4. Design Phase

With the application requirements defined, the next step was to create an architecture for the evolving implementation. The design phase provided the means to convert the user requirements into a functional architecture and component interactions. Although this project was unique in that it used a DOS-based optimizer model and displayed the output graphically using a GIS, it was similar in design to the ARIES system developed for USARC. ARIES, is an SDSS designed to evaluate and compare site desirability for Army Reserve unit locations (Murphy, 1997). This team used a Concept-to-Code (C2C) heuristic which allowed the application to be conceptualized in general terms, and then specialized architecturally around existing off the shelf components as design requirements were collaboratively prototyped (Falk, 1997).

The C2C architecture design has multiple levels of abstraction which are categorized as macro and micro. These are similar in concept to the macro and micro software development processes identified by Booch. Here, the macro architecture is representative of “what” is being designed in a general and abstracted form while the micro architecture, on the other hand, represents the “how” of the application being designed. We chose to use the “Macro/Micro” concept because it provided a clear

means by which to conceptualize the levels of abstraction when designing a system using COTS software.

This phase required several iterations to arrive at an overall architecture capable of meeting our requirements. The macro-architecture was developed quickly, followed by the U/I, and eventually the micro-architecture.

a. Macro Architecture

Developing the macro-architecture proved to be relatively straightforward. The RSLES macro-architecture diagram (Figure 4.3) followed a similar architectural design as that of the ARIES system used to evaluate Army Reserve Center locations. An attempt was made to develop an architecture in which the mapping module performed the data management functions. Our research indicated that this was indeed a viable solution, however one that could not be pursued within the time allotted for project development.

b. Micro Architecture

The choice of technologies for the micro-architecture was aided by the fact that the COTS software for two of the four modules was specified in the project requirements (MapInfo and GAMS). The choices for the data module and U/I module were all that remained. The development team reviewed current technologies and selected Microsoft's Access 97 and Visual Basic 6.0. Interoperability between components was a major concern as was the ability to manipulate databases in Visual Basic code. Using Visual Basic as the 4GL development tool, we could control MapInfo via distributed computing standards (OLE) and the Access database using the Data Access Object (DAO) 3.5 Object Model. DAO's programming interface is extremely robust and easy to use. But most importantly, with Microsoft Jet databases, DAO also allows access to features not available in SQL itself. One can also combine DAO with SQL. This was a critical factor in meeting design requirements. DAO essentially puts Visual Basic into the world of object-oriented programming (McManus, 1998). DAO has an object-oriented hierarchy that starts with the database object and its collections

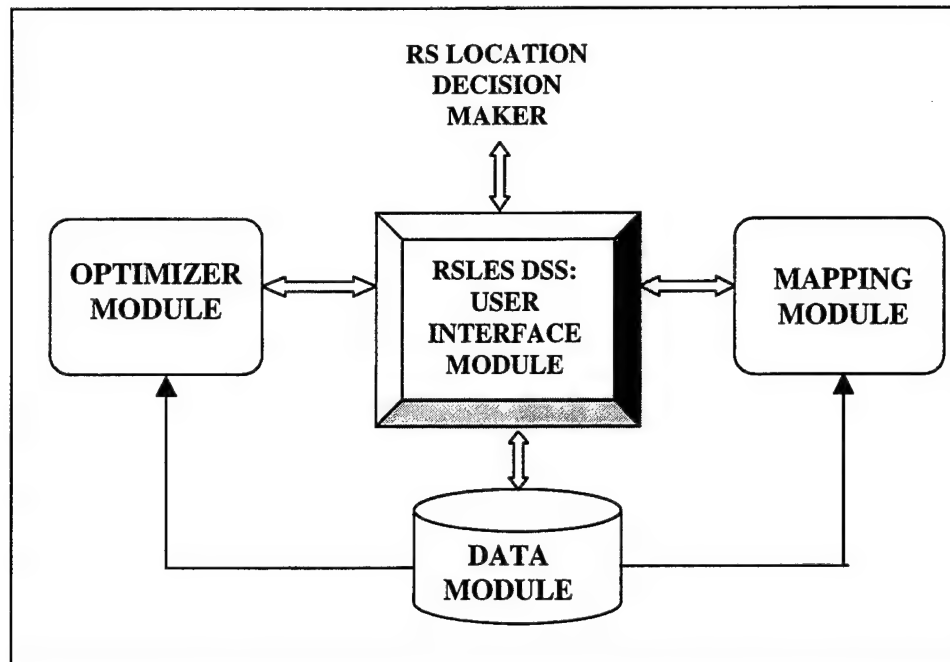


Figure 4-3. RSLES Macro-Architecture Diagram

(collections are related sets of objects). One of these collections is the Recordset, which proves to be the “workhorse” of this project, used in virtually every coding module. The resulting technologies selected to populate the macro-architecture, the reasons for their selection, and the communication protocols used are summarized in Table 4.1.

Macro-Architecture Module	Micro-Architecture Model	Reasons for Selection	Communication Protocol (OLE & DAO are Microsoft protocols)
RSLES User Interface	Visual Basic 6.0	4GL RAD Tool	OLE, DAO 3.5
Mapping Engine	MapInfo Professional 5.0	DoD Requirement	OLE
DBMS	Microsoft Access 97	Native DBMS vice ODBC	JET SQL 3.5 DAO 3.5
Optimizer	GAMS	DoD Requirement	Windows API

Table 4-1. Macro/Micro Architecture Chart

c. RSLES User Interface Design Prototype

Of the technologies selected for the micro-architecture, the NPS development team only had previous experience using Access and GAMS. Unfortunately, there was no experience using any type of external communication protocol with GAMS. Neither was there any in-depth experience for using MapInfo or Visual Basic. The first step was to obtain quality training and to become familiar with the capabilities, strengths, weaknesses, and necessary communication protocols for each of the components. The next step was to develop an initial U/I that would accommodate the necessary views while meeting minimum requirements (high priority). The “must have” items were placed at the top of our priority list while “nice to have” features were slipped to the end. The NPS development team agreed that these items would be incorporated only if the timeline permitted such during the latter stages of the project.

By using a RAD development approach, the NPS development team was able to quickly develop an interface and link the MapInfo and Access components to the U/I. This ensured connectivity with three of the four components however problems with the data sources and inability to connect to GAMS were reason for concern. A series of U/I prototypes ensued using the features of a RAD development methodology, each improving on the previous version. The following sections describe the input/output representations and control mechanisms that resulted from such an iterative approach.

(1) Input Representations and Control Mechanisms. The U/I final view includes a main view and a map view. Two views were necessary to accommodate the amount of information the user desired to view and to avoid a cluttered single view. The final U/I is shown in Appendix B (Note: The map view has several ‘popup’ screens and controls which are not shown in this view). The particular views and their controls are:

- Main View: Enables the user to select the Service, View, and Unit to display.

- Map View: Enables the user to graphically view the selected unit in its present configuration.
 - ◆ Various metrics can be selected with the results being displayed using thematic mapping.
 - ◆ Information boxes are displayed presenting various data to the user for every station associated with the selected service (District/Battalion and Zone/Company level data).
 - ◆ Button controls for *Run Optimizer*, *Display GAMS Output*, returning to the Main Menu, and clearing data input by the user. The *Run Optimizer* button allows the user to run the selected unit through the optimization model. The user has the option of identifying zip codes in which a station should be closed, a station opened, or zip codes to be candidates for either. The entered data is validated, written to files, and passed to the GAMS model. The *Display GAMS Output* button allows the user to graphically display a selected unit and view a summary report of GAMS output. The *Clear Screen* button clears any user-entered data associated with the *Run Optimizer* selections. The *Main Menu* button clears the map view and returns control to the main form.
 - ◆ A toolbar that allows the user to utilize the built-in functionality provided by MapInfo.
 - ◆ Provide a status bar to the user that provides feedback and assistance.

With the approved U/I, we moved to the Construction Phase where the appropriate code was developed for data manipulation behind the interface. This allowed us to hide the actual system implementation from the end-users, one of our primary design objectives, and permit the decision-maker to view and access RSLES from a single interface.

(2) Output Representations. The primary output of RSLES is the graphical display of results of the optimizer model. However, there are various metrics and data subordinate to the optimizer which also prove useful to decision-makers concerned with location decisions. One of the advantages of using a RAD development approach is that it allows developers to quickly converge on a design acceptable to the customer. Because customers are part of the development effort, the project has a better chance of success and can incorporate more features that the customer finds desirable,

hence leading to a higher level of customer satisfaction. Upon completion of the System Requirements Review, the following output representations were agreed upon:

- “As Is” map display. A map view of a selected unit showing locations of recruit stations, boundaries of subordinate units (if any), and zip code boundaries.
- Optimizer display. This includes a map view of the selected unit after it has been run through the optimizer model and a summary report of the displayed output. The summary includes the number of Army, Navy, and Joint stations, the number of recruiters for each Service, and the total production for each service. The user can also choose to display “pre-run” GAMS output for selected metropolitan areas of the country.
- Metrics. This view shows a thematic map display of the selected unit after the user has chosen a metric.
- Selected Unit Summary Information. Two data boxes provide summary data for any Army Battalion, Navy District, Army Company, or Army Zone. Buttons allow the user to search for a desired unit.
- Production Data. Production data for the selected unit can be displayed.

5. Construction Phase

The purpose of the construction phase is to grow and change the implementation through successive refinement, ultimately leading to the final product. The evolution of an architecture is largely a matter of trying to satisfy a number of competing constraints, including functionality, time, and space: one is always limited by the most binding constraint (Booch, 1997).

a. RSLES Development and Strategies

With the component framework established and all the required components and interfaces specified, coding activities began. Most coding activities were done in parallel starting with the *Display GAMS Output* module. Problems with input data however made this phase extremely difficult. With many data sources, it was extremely difficult to develop code when each Service used different data structures for the same item (i.e. Station ID) or worse yet, when one Service used a field that another did not.

Close coordination between the developer of the optimizer model and Visual Basic developer became critical. The project's major requirements necessitated that the script file and data files be in the proper format. Similarly, coding the application to display GAMS output required flexibility since no two outputs would likely ever be similar. An inordinate percentage of the total code was devoted to data validation and error trapping.

The ability to communicate MapBasic commands to MapInfo through MapInfo's OLE automation object provided great functionality and flexibility to the project development. Similarly, Visual Basic's ability to bind its controls to an Access database eased development of the U/I and provided the means to make quick changes when the user desires to display additional data. This complies with a basic RAD principle: it is an iterative process for demonstrating software to users as it is developed, using the immediate feedback to converge on useful solutions and minimize undesired surprises (Harris, 1997). Unfortunately, the same cannot be said for incorporating SQL into Visual Basic code. Most of the queries in the RSLES application were generated from a SQL string concatenated with user input. This proved to be troublesome in that it made coding and debugging very difficult. The alternative was to use parameterized queries, which are much faster than queries built using SQL on-the-fly in Visual Basic code (McManus, 1998). We chose not to incorporate parameterized queries because the small gain in execution time did not justify the additional effort.

b. Coding Stage

The RSLES U/I and application was coded using Visual Basic 6.0 and standard structured programming techniques. As described previously, two forms were created during the design (prototyping) phase, the initial U/I that contains Visual Basic objects and the main form that contains the integrated MapInfo object of the RSLES U/I along with several data bound controls. The remainder of the application is comprised of VBA-coded modules and a MapBasic module containing MapBasic code necessary to run MapInfo menu commands through the OLE automation object (Appendix D). The program files include:

- MainMenu.FRM - Contains procedures and logic for controlling the initial U/I
- Select.FRM - Contains procedures and logic for controlling the main U/I
- SubMain.BAS – Initializes the RSLES U/I; Loads and connects COTS components to the U/I; Displays the U/I.
- PublicDefinitions.BAS – A library module that initializes all public variables and constants.
- MapBasic.BAS - A library module that initializes all public variables and constants for communicating with the integrated mapping component.

Code development was the most difficult and time-consuming portion of the development effort. To ensure coding techniques were acceptable and to verify the code against system requirements, it was evaluated by the development team and selected NPS faculty. A combination of internal reviews, walkthroughs, and peer reviews were used. This proved to be very important since the development team itself had no previous experience in a “total” system development or with two of the applications involved in the macro-architecture.

The output displayed by RSLES is the *optimal* solution. Optimal implies a perfect scenario, where each Service has the correct number of stations and recruiters in place with which to maximize production or minimize cost. In reality however, it presents a solution that may never fully be implemented for obvious reasons. Facility rental agreements, advertising outlays, and moving expenses are but a few examples of why full implementation is impractical. What it will do, however, is stimulate the decision-maker's thought process as to why a particular station is considered optimal by the model. With this in mind, the user can use the DSS as a starting point for station location research.

The development and testing of RSLES was conducted entirely at NPS on desktop computers. While stringent evaluation and testing were not possible because of time constraints, every effort was made to meet design objectives. The ultimate test of RSLES's performance will be its usefulness in the operating environment it was designed for, not an environment such as NPS. The RSLES System Installation and System Notes instructions are located in Appendix E.

C. SUMMARY

Faced with tight time lines, RAD allowed us to quickly develop a prototype and converge on a design acceptable to the user. Once this was accomplished, we could focus on the pieces of functionality that had the highest business value, and deliver that functionality rapidly.

We sought to fix user requirements as much as possible and as soon as possible. The RAD process combats scope and requirements creep by limiting the project's exposure to change. It shortens the development cycle and seeks to limit the cost of change by incorporating as much of it as possible up-front before a large investment is made in development and testing.

THIS PAGE INTENTIONALLY LEFT BLANK

V. USE-CASE ANALYSIS

A. INTRODUCTION

The RSLES Graphical User Interface (GUI) is designed to provide functionality to two basic types of users; the joint planner and the service planner. Since there are no joint recruiting commands, a logical grouping of zip codes had to be used to allow optimization by a joint planner. The unit of aggregation we selected is the metropolitan area. In order to uniformly define metropolitan areas, we used the Metropolitan Statistical Areas (MSA) as defined by the Census Bureau. We initially identified eleven MSAs to be incorporated into the RSLES database with the intention of increasing the number to 25-50.

Since each service has its own recruiting command, a service planner would normally aggregate data by recruiting units. The U.S. Navy uses Recruiting Areas, Districts, Zones and Stations, whereas the equivalent units for the U.S. Army are Recruiting Brigades, Battalions, Companies and Stations. Table 5-1 lists the military recruiting organizational hierarchies. Limitations on the capabilities of the optimizer model restrict analysis to Battalion/District and Company/Zone levels. Since all echelons higher than the Battalion/District level are administrative, they are not included in the GUI. They contain such a large number of zip codes that aggregation and optimization is impractical.

ECHELON	Navy	Army	Air Force	Marine Corps
I	CNRC	USAREC	RS HQ/CC	MCRC
II	Areas	Brigades	Groups	Regions
III	Districts	Battalions	Squadrons	Districts
IV	Zones	Companies	Flights	Stations
V	Stations	Stations	*	Substations
VI	Recruiters	Recruiters	Recruiters	Recruiters

Table 5-1. Military Recruiting Organizational Hierarchies

B. JOINT PLANNER CASE

Joint services recruit station planning must be done at the metropolitan area level since there are no joint recruiting service units. The following is a use-case analysis of the Jacksonville, FL metropolitan area and describes how a joint planner may employ RSLES to aid in making recruit station location decisions.

1. Selecting a Metro Area

Upon initiating the RSLES application, the user is presented with the main menu. He must first select the "Joint" radio button from the "Service" selection box. Doing so enables him to select the "Metro" radio button from the "View" selection box. He may then choose the metro area he wishes to view from the drop down "Metro" selection box. The user then selects the Fiscal Year and Quarter from the appropriately labeled drop down boxes. Figure 5-1 reflects the main menu for a joint planner who wishes to view the Jacksonville, Florida metropolitan area. When the user is satisfied with his selections, he selects the "Continue" button. At this point, a message will ask the user to confirm his selection, then notify the user that the main user interface will take a few minutes to load.

The screenshot shows the 'Main Menu' window of the 'Recruit Station Location Evaluation System'. The title bar reads 'Main Menu' and the window has standard Windows 95-style controls. The main heading is 'Recruit Station Location Evaluation System'. Below it, the instruction 'Select a Service, View, Unit, Year, and Quarter' is displayed. The interface is divided into several sections: 1. 'Service' section with three radio buttons: 'Army', 'Navy', and 'Joint' (which is selected). 2. 'View' section with three radio buttons: 'Bn/District', 'Company/Zone', and 'Metro' (which is selected). 3. A 'Metro' dropdown menu showing a list of metropolitan areas: Dallas, Denver, Houston, Jacksonville (selected), Los Angeles, New York City, San Diego, and San Francisco. 4. 'Fiscal Year' dropdown menu showing '1999'. 5. 'Quarter' dropdown menu showing '2'. 6. A 'Continue' button at the bottom center. 7. An 'RSLES Controls' section at the bottom right with 'Clear Form' and 'Exit RSLES' buttons.

Figure 5-1. RSLES Main Menu

At this point, the “Select” screen, depicted in figure 5-2, presents the viewer with a generous amount of information. The left side of the screen presents unit information. The user may either scroll through the units to view information, or else designate a specific unit by selecting the search button located above or below the information box of interest. The user may also toggle between Army and Navy units by selecting the “Other Service Data” button. The major portion of this view is a MapInfo window that depicts geographically the status of the metropolitan area during the quarter and fiscal year selected. The purple boundary encompasses the zip codes that are included in the metropolitan area selected. The legend below the map depicts what the symbols signify. At this time, only green tanks, blue ships and purple stars will be shown as they represent what is currently in the RSLES database for the selected metropolitan area.

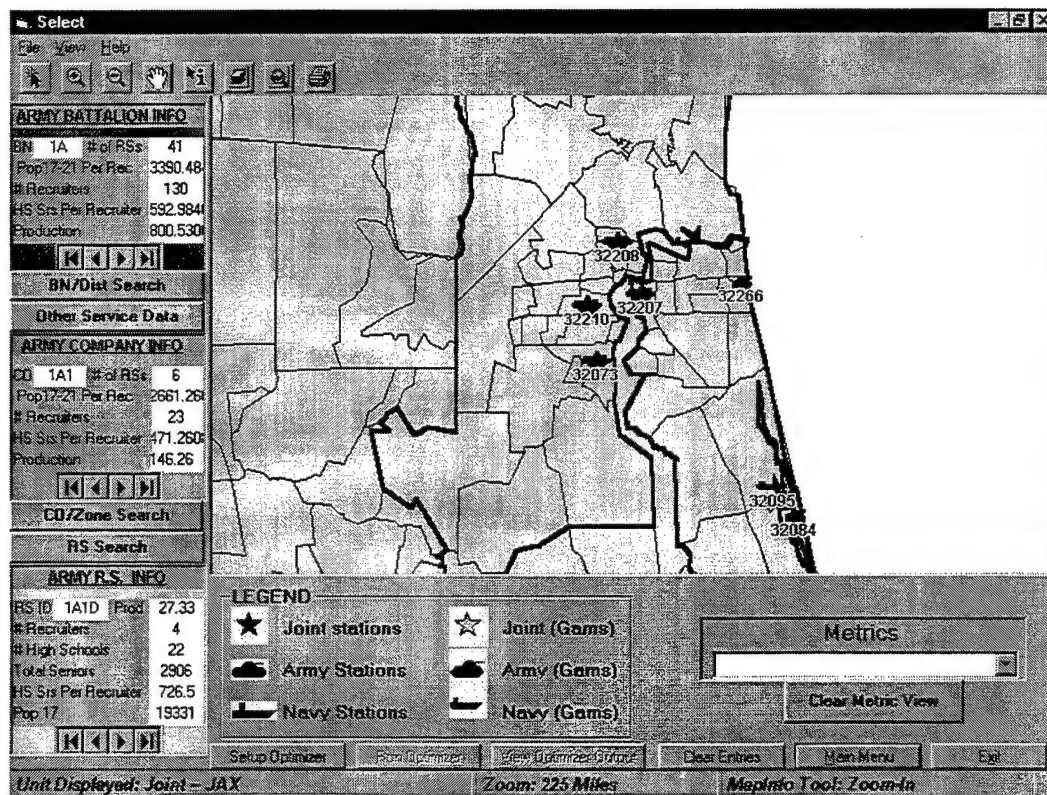


Figure 5-2. Joint User Select Box

The toolbar at the top of the screen is enlarged in Figure 5-3. It enables the user to manipulate the screen to make the data depicted easier to understand. The button with the



Figure 5-3. MapInfo Toolbar

arrow is the selector button. When activated, the cursor turns into an arrow. When held over an object, a bubble box will appear that identifies the object. The buttons with magnifying glasses are for zooming in or out on the map. The button with the hand on it is the “grabber” button and it allows the user to slide the map in any direction to view portions of the map that are beyond the screen viewing area. The “i” button will change the cursor into a cross and will provide further information on any object that is selected with it. The button to the right of the “i” button is the layer control button. It offers the user the ability to change the order in which MapInfo maps are layered. It also allows the user to add labels and text to the map image. The button with the magnifying glass and the blue check resets the map to the original state it was in when it first appeared. The printer button is self-explanatory.

2. Viewing a Thematic Map

Selecting an option from the “Metrics” drop-down box in the lower right corner of the “Select” screen activates the RSLES thematic mapping feature. The “Metrics” drop-down box is enlarged and depicted in Figure 5-4. Thematic mapping allows the joint user to geographically display zip-code level data in a way that makes it easy to compare selected metrics of each zip code against others within the selected metropolitan area. Selecting the “Metrics” drop-down box will list the three options currently available to the user.

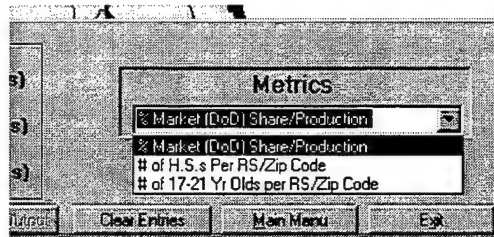


Figure 5-4. RSLES Metrics Drop-Down Box

The first option is titled “% Market (DoD) Share/Production per Zip.” For the Joint Planner, this option thematically maps the selected metropolitan area by color coding each zip code based on the annual average of DoD accessions obtained from that zip code buffered over three years.

The second option in Figure 5-4 is “# of HSs per RS/Zip Code.” For the joint planner, it color codes all the zip codes in the metropolitan area based on the number of high schools in that zip code. The third option is “# of 17-21 Yr Olds per RS/Zip.” This option color codes zip codes by the number of high schools within the zip code. Figure 5-5 depicts the result of selecting the percent market share metric. The resulting thematic map paints zip codes with 0-5 DoD accessions per year red and zip codes with more than 20 DoD accessions per year green with gradations of colors for zip codes with DoD accessions in between. The legend provides details of the significance of the colors of the zip code.

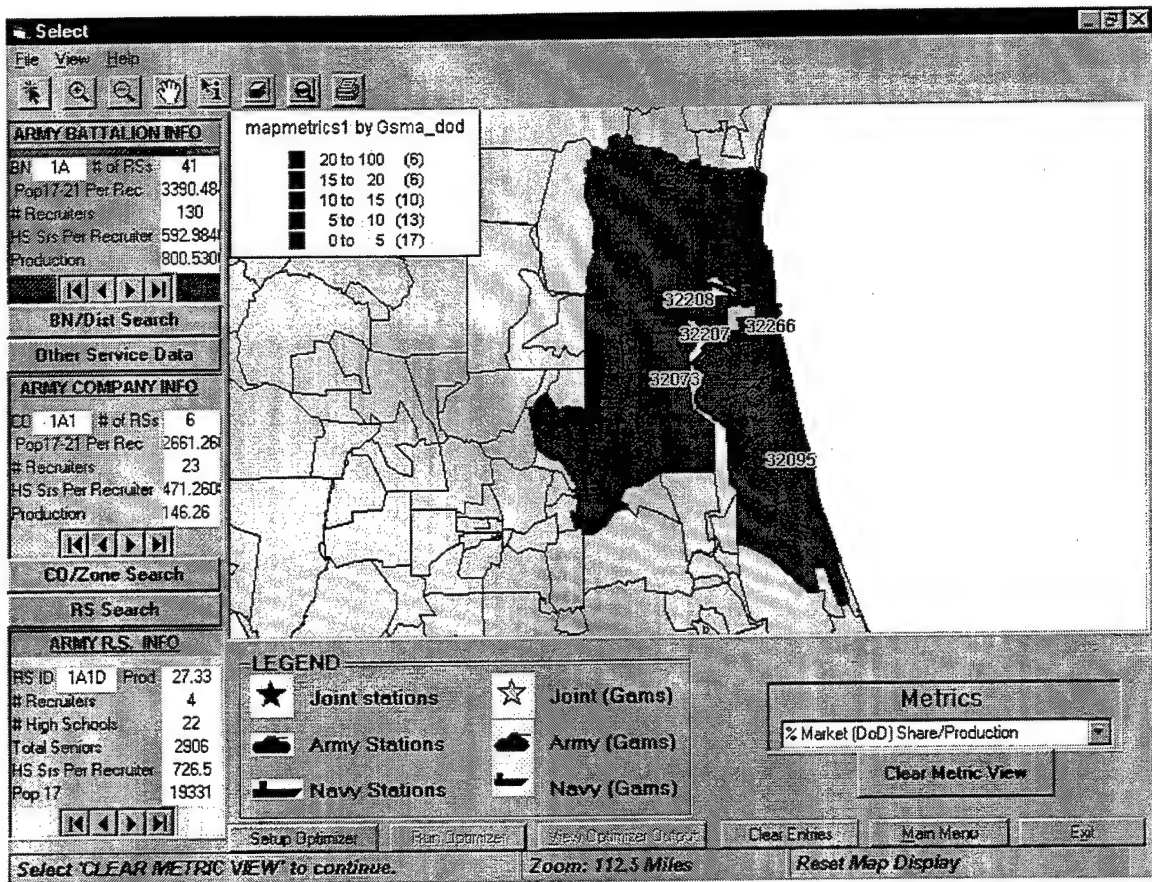


Figure 5-5. Joint User Thematic Map

3. Using the Optimization Model

Once the joint user has finished looking at a thematic map, he selects the “Clear Metric View” button to return to the previous view. At this point, he is ready to run the optimization model. To do so, he selects the “Setup Optimizer” button. This presents a view that prompts the user for input, as depicted in Figure 5-6.

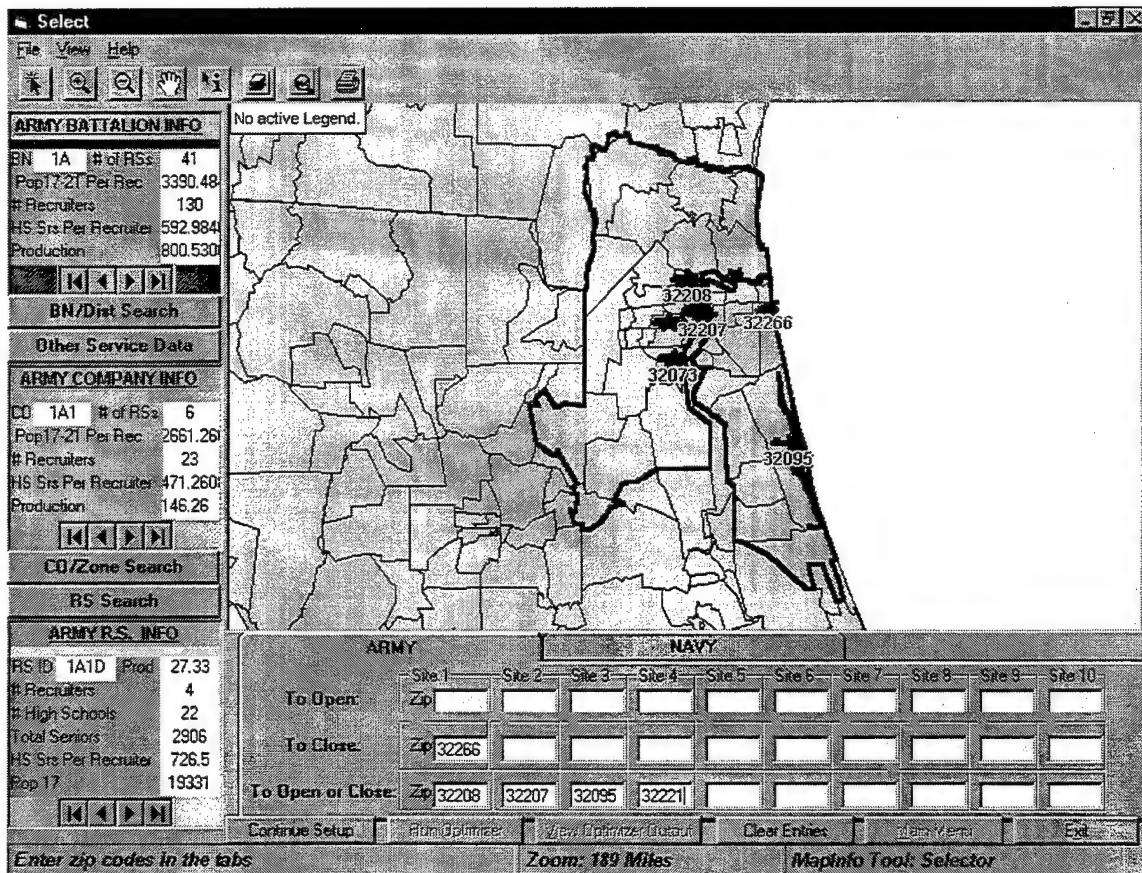


Figure 5-6. Joint Planner Optimization Screen

a. Selecting Zip Codes to Open, Close or Free

Below the map, the joint user selects the service he wishes to work with first. In the case depicted in Figure 5-6, the user selected Army. He may designate certain zip codes in which he definitely wants to open an Army station by typing them into the text boxes on the first line, labeled "To Open." In the case in Figure 5-6, the user does not wish to *require* any specific zip codes to have Army stations opened in them. Note: an error will be returned if the user attempts to enter a zip code where a station should be opened but which presently has an Army station in it.

The next step is to determine which Army Stations the joint user definitely wants to close. He does this by entering the zip code of each station to close in the text boxes on the second line labeled "To Close." In this case, the user wishes to close the

Army station located in zip code 32266. An error will be returned if a user attempts to enter a zip code where a station should be closed and a station does not currently exist there.

The next step is to enter zip codes that the user wishes the optimization model to consider for opening or closing into the third line, labeled "To Open or Close." In effect, the user is removing all constraints from the optimizer regarding these zip codes. These zip codes, may or may not have stations in them.

After filling in the zip codes in these three lines, the user selects the Navy tab and repeats the process for Navy stations. Any zip code that is not entered under the Army or Navy tabs will be addressed as follows: If the zip-code currently does not have a station in it, it remains without a station; if it does have a station in it, that station will remain. The optimization model will also consider changing the number of recruiters assigned to each station to optimize production.

b. Initializing GAMS

After designating zip codes to open, close or free for each service, the user then clicks the "Continue Setup" button at the left lower edge of the screen. At this point, RSLES conducts a query against the database for the total production of each service within the metropolitan area. It then displays a dialogue box as depicted in Figure 5-7.

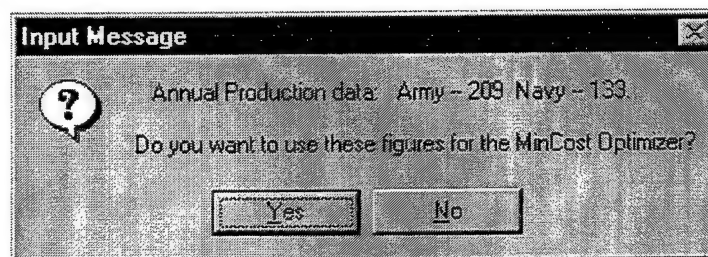


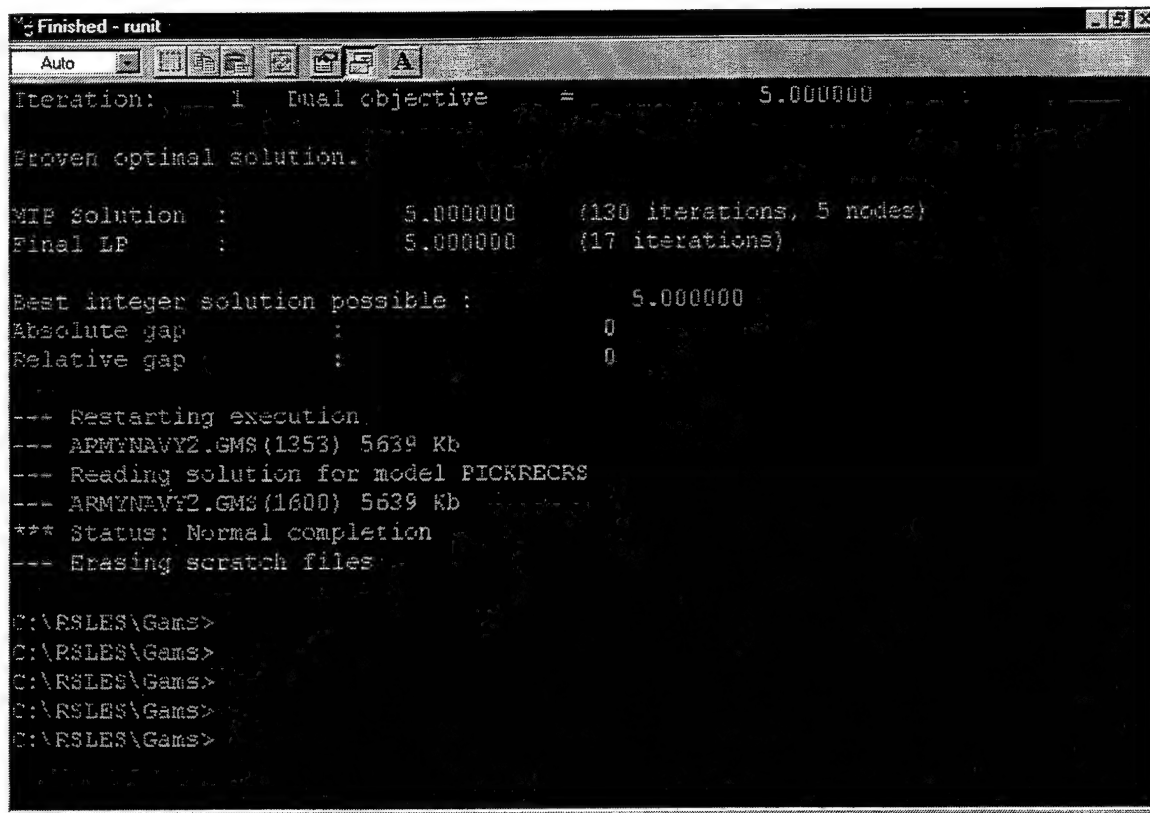
Figure 5-7. Joint Production Dialogue Box

The dialogue box in Figure 5-7 displays each service's annual average production averaged over three years for all the zip codes in the chosen metropolitan area

as of the quarter chosen in the main menu. RSLES will use that figure as the production goal for the min cost optimization model. The dialogue box will ask the user if he wishes to change either production goal. He may then do so, if desired. Once the production goal is established, RSLES is ready to run the optimization model. To do so, the user clicks the "Run Optimizer" button. At this point, RSLES generates the text files necessary to run GAMS from the user input. These text files are copied to the RSLES directory, and GAMS is executed. A DOS shell window is opened and the user can view the GAMS messages as the optimization model is run. By viewing the messages, the user can determine if the model reaches a feasible solution. There are several reasons why a feasible solution may not be reached. For example, if a station in a highly productive zip code is required to be closed and a two sparsely populated zip codes are freed to be opened, the model may not be able to achieve the production goal desired within the constraints established by the user.

c. Recovering GAMS Output

When the optimizer model successfully completes its processing, the DOS window will appear like the one depicted in Figure 5-8. The user must close the window by clicking on the 'X' at the upper right corner of the DOS window. It is important to view the DOS window to ensure that an optimal solution was achieved before continuing.

A screenshot of a DOS window titled "Finished - runit". The window has a standard DOS interface with a menu bar (Auto) and a toolbar. The text inside the window displays the results of a GAMS optimization run. It shows that the dual objective is 5.000000, a proven optimal solution was found, and the MIP solution is 5.000000 (130 iterations, 5 nodes). The final LP is also 5.000000 (17 iterations). The best integer solution possible is 5.000000, with both absolute and relative gaps at 0. The window then shows a restart of execution, reading solutions for models PICKRECRS and ARMYNAVY2.GMS, and a status of "Normal completion". Finally, it shows the erasing of scratch files and the command prompt at C:\RSLES\Gams>.

```
Iteration: 1 Dual objective = 5.000000
Proven optimal solution.
MIP solution : 5.000000 (130 iterations, 5 nodes)
Final LP : 5.000000 (17 iterations)
Best integer solution possible : 5.000000
Absolute gap : 0
Relative gap : 0
--- Restarting execution.
--- ARMYNAVY2.GMS(1353) 5639 Kb
--- Reading solution for model PICKRECRS
--- ARMYNAVY2.GMS(1600) 5639 Kb
*** Status: Normal completion
--- Erasing scratch files
C:\RSLES\Gams>
C:\RSLES\Gams>
C:\RSLES\Gams>
C:\RSLES\Gams>
C:\RSLES\Gams>
```

Figure 5-8. GAMS Interface for Successful Optimization

After closing the DOS window, RSLES will then ask the user if the optimizer found a feasible solution. If it did not, RLSES will offer some tips on how to re-define the problem to achieve a feasible solution and allow the user to try another optimization run. If a feasible solution was achieved, RSLES will graphically depict the solution as in Figure 5-9.

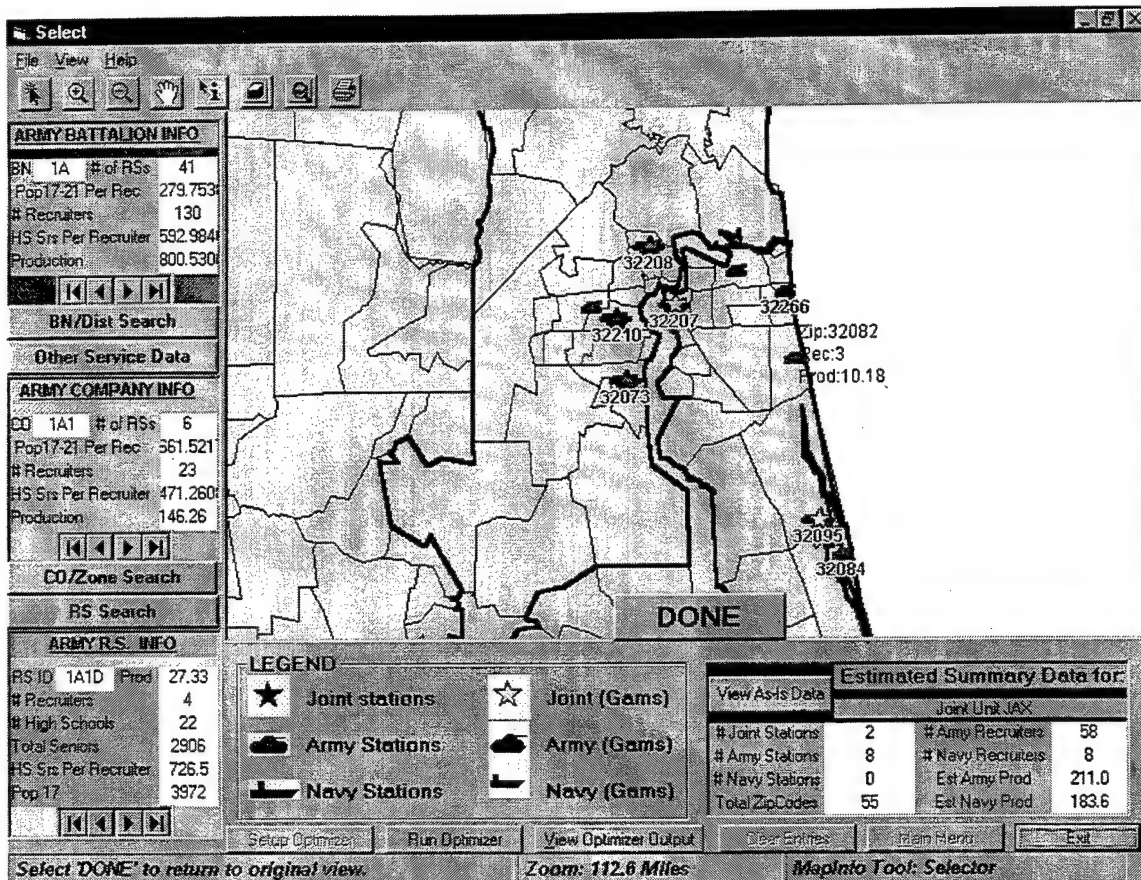


Figure 5-9. Joint Service Optimization View

The GAMS output will overlay the "As is" geographical depiction of the metropolitan area from Figure 5-2 with the symbols on the right side of the legend. Any existing station that does not have a GAMS symbol over it is recommended for closure. Similarly, zip codes with a GAMS symbol that previously had none are recommendations for opening, and stations with GAMS symbols overlaid are recommendations for leaving as is.

At the lower right corner of the screen RSLES displays the total number of Army, Navy and Joint stations the optimizer recommends along with the total number of recruiters for each service. It also displays the estimated production for each service within the metropolitan area. The user may toggle to see the same type of information with respect to the current situation in the metropolitan area. This feature

will enable the user to see what changes GAMS recommends, and how these changes are estimated to effect production.

C. SINGLE SERVICE PLANNER CASE

Although RSLES was commissioned by OSD for use by joint service planners, it was also designed to be a tool for single service planners who wish to model how changes within their recruiting units will affect accessions. Currently, restrictions on how many zip codes may be included in the domain of the optimization problem limit use of RSLES to the Company/Zone and Battalion/District levels. A single service user would use RSLES in much the same way a joint user would. However, RSLES imposes limitations that only allow manipulation of his own service's recruiting stations. Instead of using metropolitan areas, which may cross unit boundaries, RSLES allows the single service user to evaluate the station locations within recruiting units.

1. Selecting a Unit

The single service user would also begin a RSLES session from the Main Menu Screen in Figure 5-10. However, once he selects his service, the Metropolitan option as well as all options for the other service become disabled to him. In Figure 5-10, the planner is evaluating the recruit station location for Army Recruiting Company 1A4.

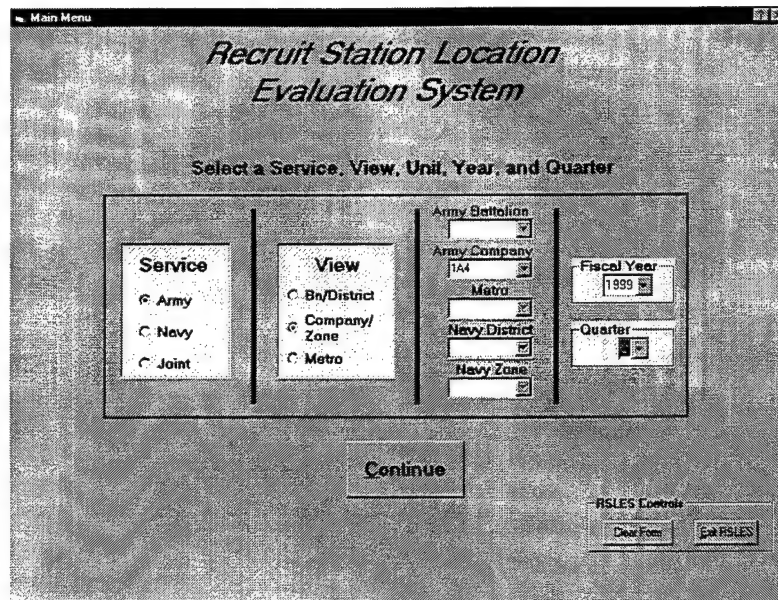


Figure 5-10. Main Menu for Army Planner

After indicating his service and unit, the planner will see the select screen. It will display the current arrangement of recruiting stations in and around the selected unit with the recruit station boundaries. Although the user may not manipulate Navy recruiting stations, they are still visible to him to provide information he may need in his decision making process. Figure 5-11 displays his selected view.

All of the functionality of the select screen available to the joint planner is also available to the single service planner. The primary difference in the view is in the boundary layer. The Army service user will see individual zip codes bounded by thin black lines, and the zip codes aligned with each Army RS bounded by thicker brown lines.

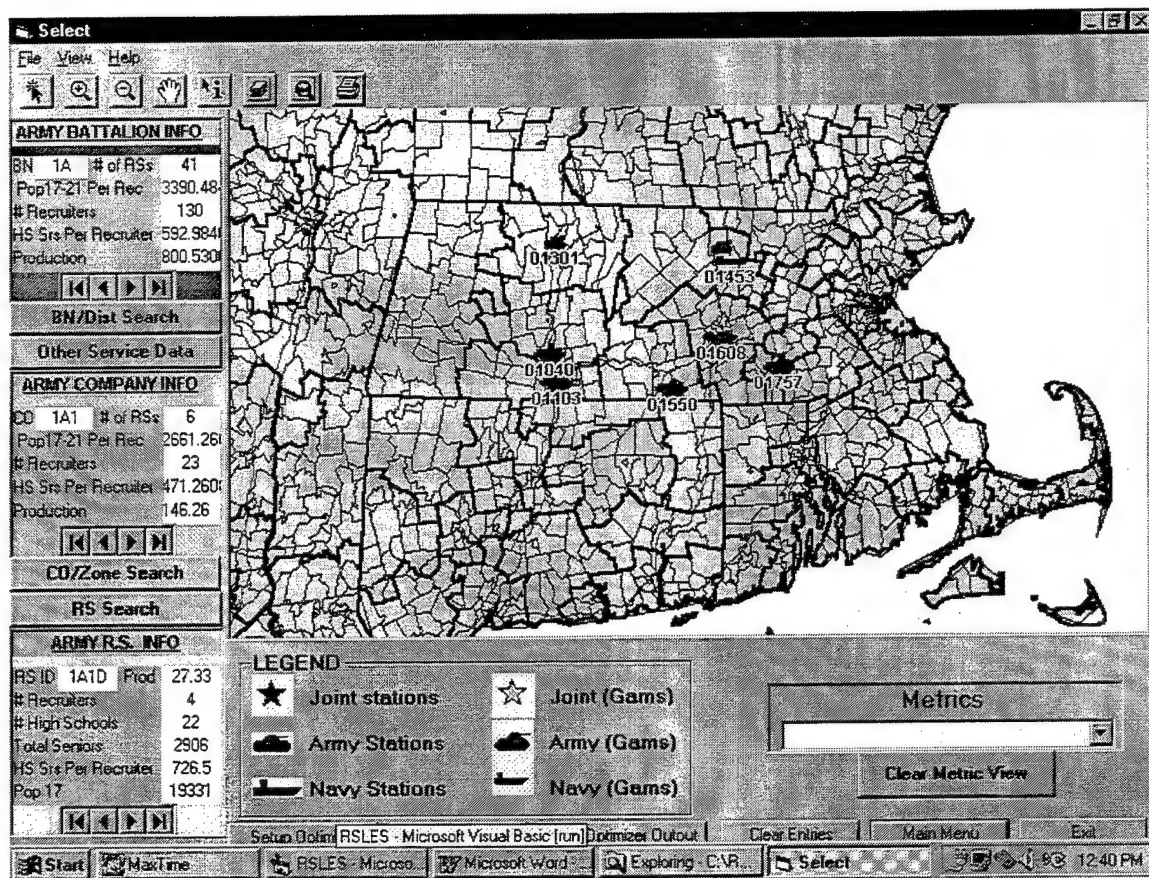


Figure 5-11. Select Screen for an Army Company

2. Thematic Mapping

The single service user may also utilize the thematic mapping option available to the joint service planner. In this case, the information is mapped at the RS level rather than the zip code level. Figure 5-12 depicts each Army RS region thematically mapped by market share. Market share is defined as the total production of all the zip codes in that RS' region divided by the total DoD production for those same zip codes. The colors range from red at the bottom of the legend to green at the top. The user can quickly determine how well each station is doing compared to the other stations in the Company and against the other services in the same area.

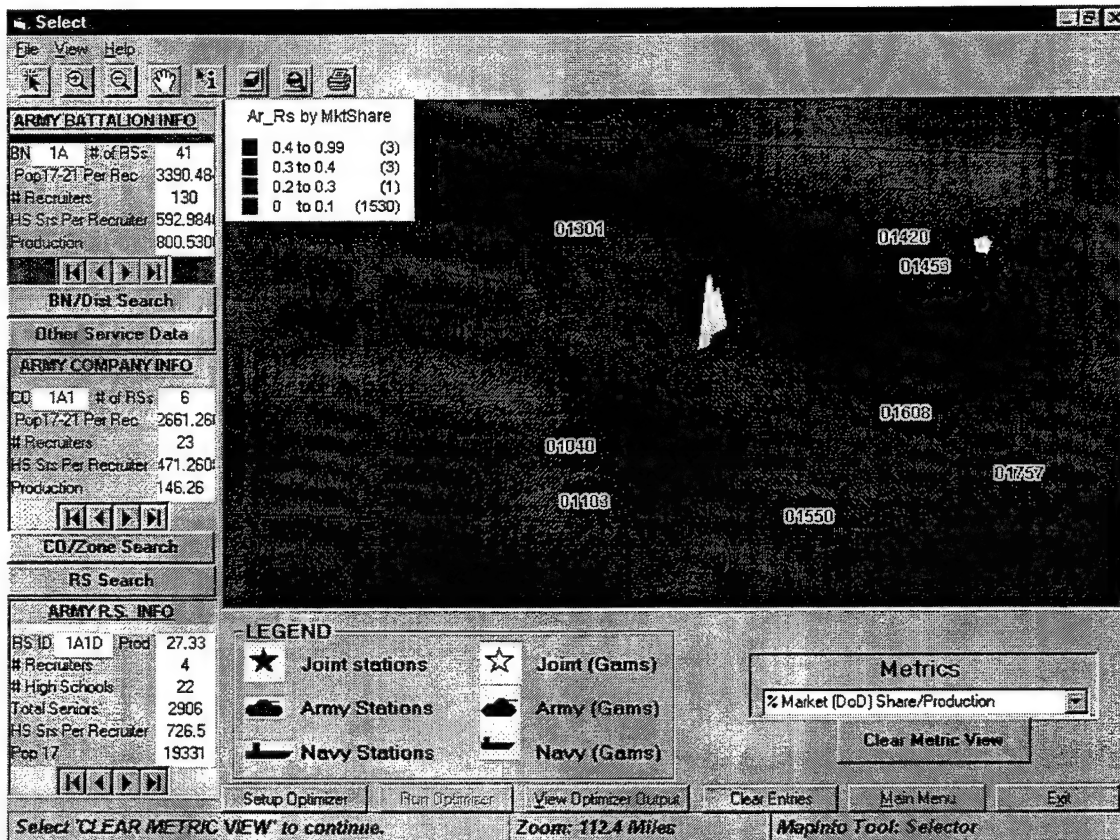


Figure 5-12. Thematic Map of an Army Recruiting Company

3. Optimizing a Recruiting Unit

The optimization feature for the single service planner works much like it does for the joint planner. The significant difference is that the single service planner is not allowed to manipulate the recruiting station locations for any service but his own. Figure 5-13 depicts the optimization setup screen for the Army planner. Notice, that he does not even have a Navy tab.

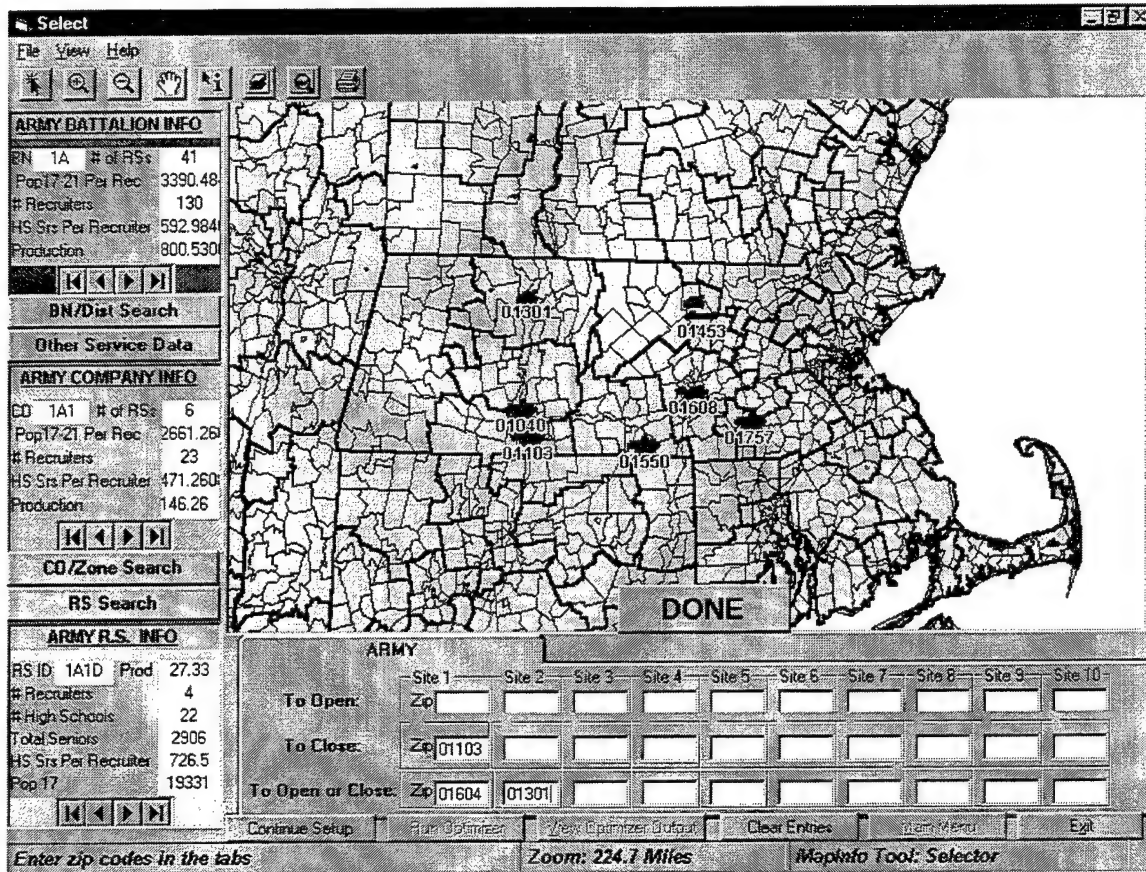


Figure 5-13. Army Optimization Model Setup

In Figure 5-13, the Army planner has decided to find the optimal solution which closes the station located in zip code 01103, and considers opening a station in either zip code 01604 or 01301. The user then clicks on "Continue Setup" to generate the text files required to run the GAMS optimizer. RSLES will compute the total production for Recruiting Company 1A4 and generate a dialogue box (Figure 5-14) that asks the planner if he wishes to use that number for the production goal of the optimization model. In this case RSLES computes the annual production for Company 1A4 to be 140. (RSLES truncates any fractional value), and the user opts to use the same value. If mission had changed, or the Company failed to make mission, the user may opt to change the value.

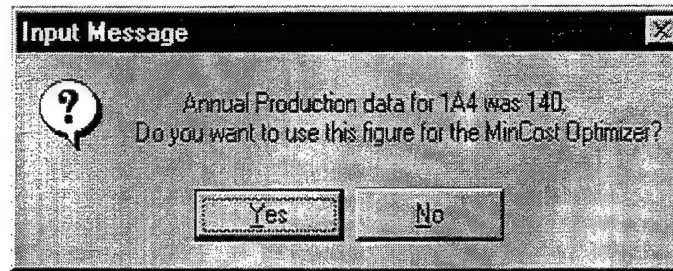


Figure 5-14. Production Goal Dialogue Box

After answering several dialogue boxes that attempt to ensure he has made all the entries he wants to make, the user will click on the "Run Optimizer" button. At this point, a DOS shell will appear and the user may view the GAMS interface as it solves the problem. Again, a successful optimization will generate a screen as shown in Figure 5-8.

After answering the error handling message boxes, the user clicks on the "View Optimizer Output" button to see a geographical depiction of the optimizer recommended solution. Figure 5-14 is the output message for Army Recruiting Company 1A4. As with the joint planner use-case, any Army station represented by a green tank not overwritten by a brown tank is recommended for closure. The RS at zip code 01301 falls into this category. The optimizer also recommends removing the Army recruiter from the joint recruiting station in zip code 01103. This is indicated by the purple star being overwritten by a red ship. Absence of a yellow star indicates that the station is no longer joint, but rather a single Navy station.

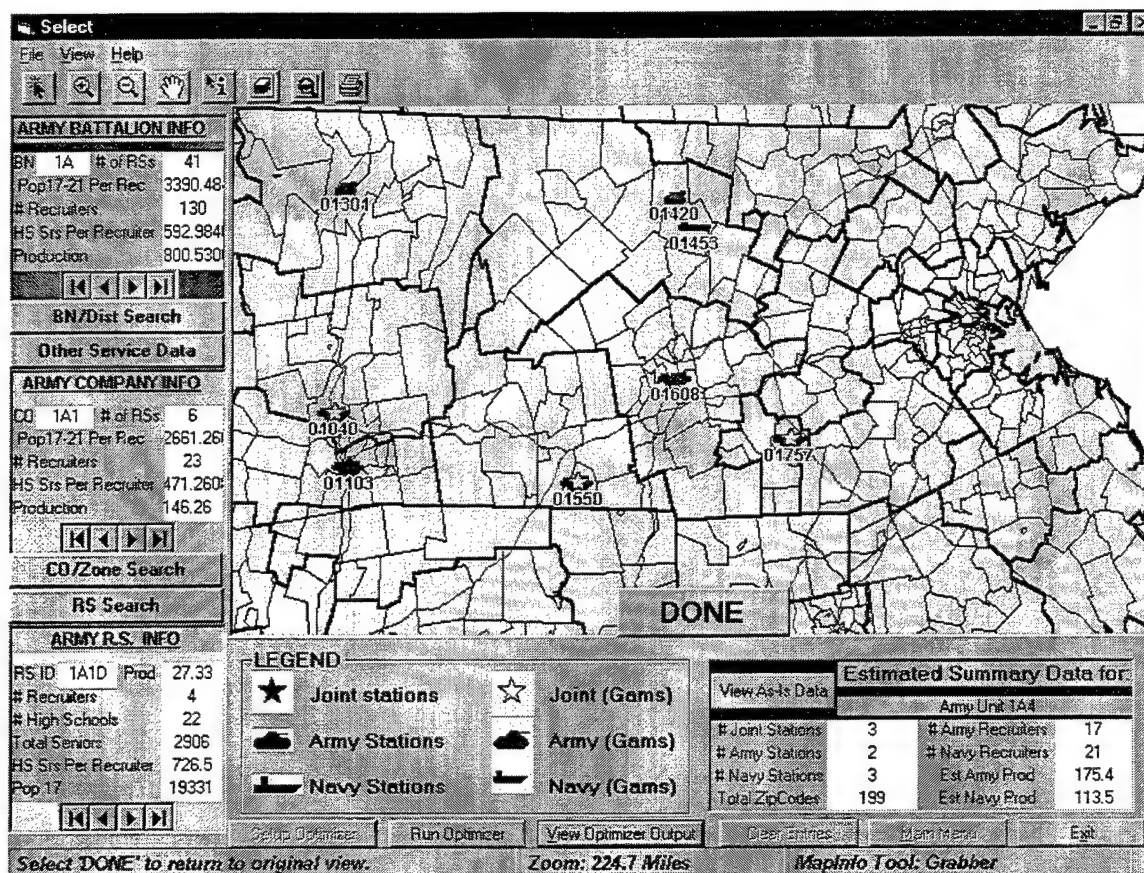


Figure 5-15. Geographical Optimizer Output for Company 1A4

The "Estimated Summary Data for:" box provides summary data regarding the optimizer output. It totals the recommended number of Joint, Army and Navy stations as well as the total number of recruiters for each service within the geographical area covered by Company 1A4. It also furnishes the estimated production of each service for this solution. By clicking on the "View As-Is Data" button, the user may toggle back to the "real world" data and quickly see what changes the optimizer recommends.

D. SUMMARY

The primary purpose of RSLES is to provide OSD and JRFC with a decision support system to assist joint recruit station planners in making location decisions. However, we decided that RSLES could also be beneficial to the service recruiting commands if it could provide a service specific view of the underlying data. This was

accomplished by creating a main menu which asks the user to identify whether he is a joint planner or a service planner. This decision determines how RSLES will group and present zip codes for viewing and optimizing. Under the joint view, zip codes are grouped by metropolitan areas. From the single service view, they are grouped by the user's choice of battalion/district or company/zone depending on the specific service selected.

The joint and single service views behave almost identically apart from the difference mentioned above. In both cases, when running the optimization model for a unit or metropolitan area, it is essential to view the GAMs output in the DOS window before continuing to ensure an optimal solution was found. Failure to do so will cause RSLES to search for output files that do not exist and an errant condition to occur.

In either the joint or single service view, RSLES enables the decision maker to graphically view the current situation, pose a series of "what if?" questions and view the results in both graphical and tabular formats. Through thematic mapping, RSLES presents selected metrics in a way that enables the user to see patterns that might not be as apparent if presented in a table.

We have just discussed how RSLES interacts with the user to facilitate decision making by graphically portraying the existing data. In the next chapter we will discuss problems encountered with the data, how those problems were overcome, and recommendations for future enhancements to RSLES.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. DATA QUALITY AND FUTURE ENHANCEMENTS

A. INTRODUCTION

When we decided to undertake RSLES development for our thesis, one of the initial attractions was that the data was already available for the project. However, soon after obtaining this data, we realized that it was far from accurate. In fact, the vast amount of data and the extent of its inaccuracies made it eventually more cost efficient to generate new datasets than to try to correct that which was already available. This chapter will address the quality of the original data, how anomalies between sources for new data were addressed, and recommendations for future developments to RSLES to maintain data quality and to migrate to a Web-based environment.

B. DATA ANOMALIES

1. Initial Data Errors

The initial dataset from USAREC was stored in Statistical Analysis System (SAS) format. While converting this file into a Microsoft Access Database, we discovered that approximately 8% of the zip codes were duplicated yet had different associated data values. Then, utilizing the MapInfo geographical coding function, we mapped the data to the United States by Army Recruiting Brigades. This mapping revealed that at some time during the preparation or handling of the database, 1874 zip code level records from the New England area had had their zip codes stripped away and replaced with one from some other part of the country. The graphic result was that over 30% of the zip codes in the Army's 1st Brigade seemed to be randomly scattered across the country as depicted in Figure 6-1. The associated database table created the illusion that zip codes could be used more than once in different parts of the country. This revelation made it obvious that new, clean data had to be collected in order for RSLES to realistically present the current situation and to generate accurate input files for GAMS.

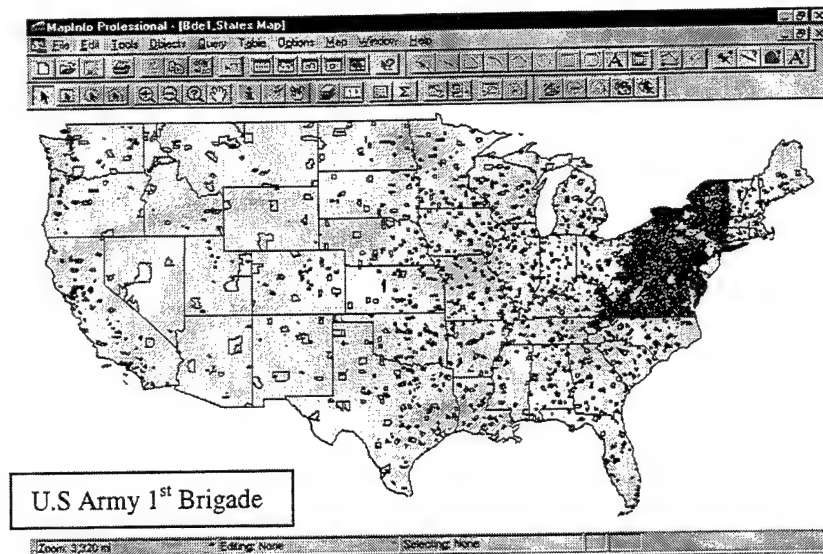


Figure 6-1. Mapping of Flawed Data

2. Anomalies Between Data Sources

While combining the data from the different sources several anomalies were discovered which required assumptions to be made. The first anomaly involved the domain of zip codes used by each source. There are nearly 50,000 zip codes in the United States and the number changes frequently. The ATAS data furnished information for nearly 43,407 zip codes. However, the CNRC data only furnished information for slightly more than 40,475 zip codes. After eliminating all the zip codes associated with Army Post Office (APO) and Fleet Post Office (FPO) addresses, as well as those zip codes in U.S. territories, the ATAS domain was pared down to 39,612 zip codes, while the CNRC data had 39,714. This difference represents a 0.26% error and may be attributable to differences in the currency of the zip code information used by USAREC and CNRC. In order to avoid null fields, these 102 zip codes were dropped from the database. This action has a minimal impact since none of these zip codes had recruiting stations within them. Furthermore, there was no population or accession data available for these zip codes.

The Navy data also had anomalies that first called to question its integrity. Cross-referencing the two tables provided by CNRC resulted in the following inconsistencies:

- Numerous stations had no recruiters assigned.
- Numerous stations were not located in zip codes that were assigned to them.
- There were several stations that had no territory assigned to them.

These anomalies contradicted assumptions that were made based on the ATAS data that was built into the logic of the RSLES application. In the first instance, computations to determine metrics such as population of 17 to 21 year olds per recruiter will result in a 'divide by zero error'. The second anomaly does not represent a problem unless two stations are within the same zip code. Both the RSLES GUI and its underlying optimization model identify recruiting stations by the zip code they reside in. As such, RSLES will treat two separate stations within the same zip code as one. For example, two services operating recruiting stations in different buildings within the same zip code will be reflected as a joint recruiting station in the RSLES display. The third anomaly is rare and of little significance.

Another anomaly, separate from those encountered with CNRC data, was the result of using the Census Bureau's data for designation of metropolitan areas. A mapping of the Census Bureau information revealed numerous holes within the metropolitan areas (Figure 6-2).

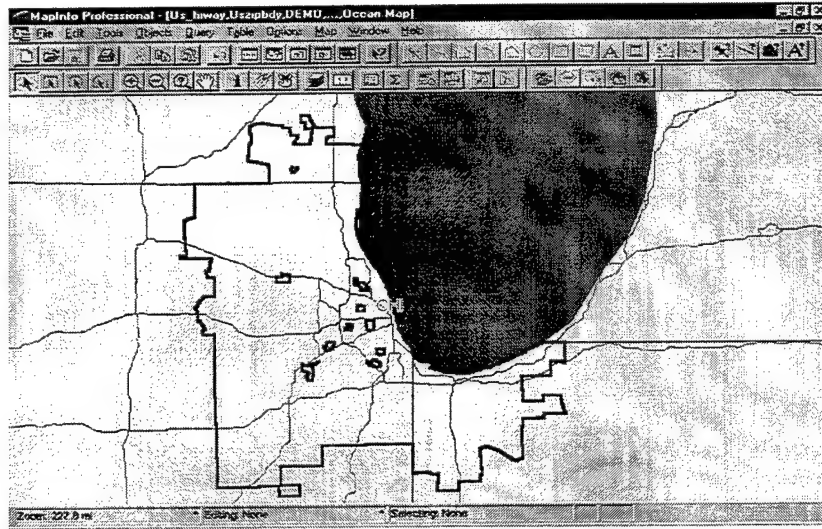


Figure 6-2. Chicago Metropolitan Area with missing zip codes

Most missing zip codes could be located in the RSLES database and assigned the metropolitan designator for the area in which they appear to be located. The resulting correction provides a more homogenous looking metropolitan area (Figure 6-3), however, not all 'holes' could be removed from most metropolitan areas. Those remaining zip codes are the result of the difference between the RSLES database and the MapInfo Zip Code tables. These differences are minor and should have minimal impact on the functionality of RSLES.

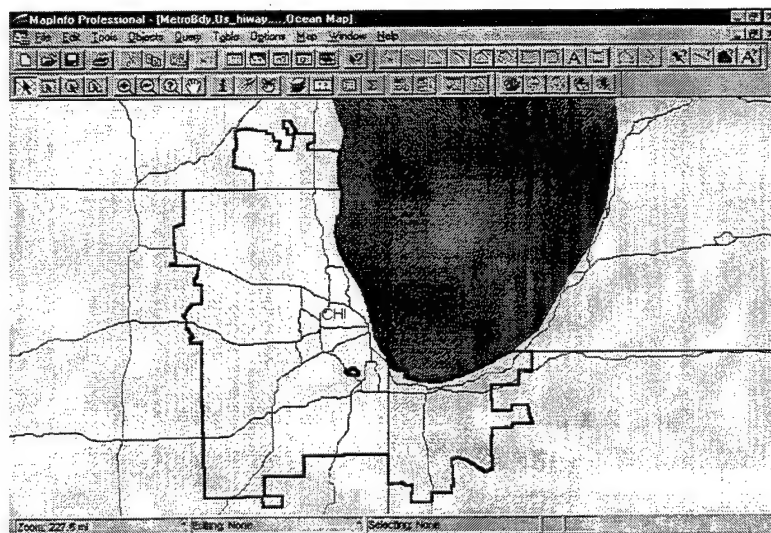


Figure 6-3. Modified Chicago Metropolitan Area

C. FUTURE RSLES DEVELOPMENT

The primary benefit of the RSLES project is the use of a very powerful decision tool that considers all possible combinations of recruit station locations for both services and provides the user with the optimal solution. Previously, this capability was only available for a single service. By graphically displaying the results of the optimization model, recruit station planners may benefit by being able to better analyze relocation decisions in both intra-service and cross-service modes. The RSLES application goes far beyond mere data display, allowing the decision-maker to manipulate location decisions in a highly flexible and fully functional decision environment.

1. Data Warehouse

Because of the complexity of the queries, the business rules and the quantity of data involved, the development team identified a critical need for a centralized data resource file. A future version of RSLES should employ data warehousing techniques to conduct extractions from the many disparate source data files from each of the services and government agencies, and 'cleanse' it for the RSLES model. Since RSLES deals with data of a spatial nature, basic data warehousing techniques, such as aggregation and summarization can be used to take advantage of these spatial attributes of the source data as well as other attributes. The spatial aspects of the data can then be used in conjunction with the GIS application to maximize query performance and evaluate the data by permitting a wider variety of metrics to be developed. Another advantage of using spatially identified information in queries is a significant increase in performance, since this approach narrows the scope of the query to a specific area under consideration.

2. Data Administrator

The additional time required to collect the data from all the different sources mentioned in Chapter II, to merge them into the database, and to check for accuracy detracted from the time available to make additional refinements to the RSLES GUI. Like any other DSS, RSLES depends on quality data and the ability to maintain that data. Unfortunately, time did not permit the development of a data administrator for this

project. An important element in establishing and maintaining an expected level of quality for data is the assignment of individual service responsibility for identifying and correcting the inadequacies of the source data files. Further data quality analysis will provide the Directorate for Accession Policy (AP) office with the assurance that the RSLES application is basing its outcome on valid data that is accurate, consistent, complete, timely, and unique. A data administrator will provide a stable data set to the RSLES application by using queries that can be executed repeatedly even as the source files change. The administrator can also provide an automated data quality filter that facilitates data cleansing of the source data sets. A data administrator application should be developed using Visual Basic™ for easier integration with RSLES. The administrator would act as a migration engine to transform the source data into the structure required by the RSLES application. Developing a mechanism by which to provide the DSS with high quality data creates a foundation for a much higher probability of successful implementation of decision-based systems.

3. Migrating RSLES to the Web

Even though RSLES was designed to run as a standalone application, its future lies as a thin client in a three-tiered architecture. This would allow most if not all of the processing to be done on one or more servers. Moving to such a distributed, partitioned architecture would of course require significant reengineering of the application. When combined with migration to the Web, the benefits of less maintenance and improved functionality can be realized, as well as reduced cost. Administrators may then update the Web pages that run the RSLES application, and users can access them automatically using their browsers.

Additionally, since RSLES was designed to function as a standalone application with a heavy reliance on Data Access Objects (DAO), SQL commands must be processed on the client computer, which is much less efficient than processing on a server. Visual Basic 6.0 has remote data objects (RDO) which are specifically designed to support a client/server relationship. RDOs are an object-oriented way to access client/server data sources, that allows access to remote data in a Visual Basic application. For example, in DAO with Jet, all queries are processed on the local computer, which is generally less

efficient (McManus, 1998). The rdoQuery object, on the other hand, lets SQL commands be generated on the client and passed to the server where they can be executed with greater speed. Accessing data using the remote data control to access client/server data is an enhancement that should be incorporated into the next version of RSLES.

Another good reason to use RDO is platform independence. Because RDO is designed to be a thin layer around the ODBC API, it is capable of bypassing the Jet database engine entirely. It also supports features such as establishing asynchronous connections, batch updates, and others that go well with client/server systems (McManus, 1998). This capability will enable RSLES to run at a consistently high level of performance regardless of the platform utilized by the client.

Another feature of Visual Basic 6.0 is its support for ActiveX Data Objects (ADO). ADO allows the user to access data from a Web server in the context of a Visual Basic application. Since ADO is provided in the form of an ActiveX server library (like DAO and RDO), it can also be used by Visual Basic applications. Its performance rivals that of RDO (McManus, 1998), which means that RSLES can be "web-enabled" fairly quickly and easily from within the 4GL environment. With many DoD networks operating through NT servers, the "web-enabled" implementation of RSLES could more fully take advantage of the infrastructure available in recruiting commands such as CNRC and USAREC.

Ideally, in this "thin client" architecture, GAMS would reside on the server side. This arrangement would decrease the processing time required by GAMS and save money by eliminating the requirement to license GAMS on all client computers running RSLES.

Finally, Visual Basic 6.0 has another new feature called Internet Transfer Control, which lets the user transfer files easily to and from the Internet. This would be a useful feature when combined with the data administrator function. Each of the services could download the latest station recruiting information from the Internet into the data administrator through the Visual Basic interface. This would provide the most timely and accurate data possible to users, thereby allowing them to make location decisions based on the most recent data available.

C. CONCLUSION

Our initial estimates of the time required to develop RSLES were based on the naïve assumption of clean, available data. Once this presumption was proved incorrect, it became necessary for one of the team members to devote the majority of his time and energy toward database development, while the other focused on Visual Basic application writing. The need to devote so much effort to database development led to a concomitant increase in the amount of time required for the completion of the overall application.

While RSLES, as is, fully meets all the primary requirements assigned in the Requirements Definition phase and those additional requirements that surfaced throughout the development cycle, it still lacks some functionality that would make it a very desirable product for the joint recruiting community. These include an automated data warehouse and administrator, and a migration to the Worldwide Web.

VII. CONCLUSIONS

A. INTRODUCTION

This chapter summarizes the issues and ideas that emerged during the development and implementation of the RSLES application. The RSLES prototype application has proven the viability of the RAD development paradigm for speedy application development. It also demonstrates the robustness of using a 4GL-development tool such as Visual Basic to integrate various COTS software components into a DSS. As a result of this effort, decision-makers in the recruiting community will now be able to leverage station data to make more informed decisions regarding future station location.

B. CONCLUSIONS

This research implemented an emerging component-based methodology to create an integrated application in support of complex recruit station-location decisions. As a proof of concept application, RSLES demonstrates the ability to integrate a GIS mapping engine, a DBMS, and a powerful model solver in a seamless and flexible environment that allows users to leverage operational recruit station database information for decision-making purposes.

The development team's lack of prior experience with any of the development tools used in this project demonstrates the validity of using component-based computing, in conjunction with rapid application development (RAD) techniques, to quickly create a usable decision support application. Despite our lack of experience, we were able to apply the techniques associated with RAD to develop a successful application in a relatively short period of time (7 months elapsed time). The project started with some quickly conceived, simple screen displays developed in Visual Basic based on generic requirements for the system and what the developers thought the users would like to see in the user interface. From this, a mock-up user interface was built in a relatively short

period of time. Based on this prototype, the user could now envision what requirements were desired in later versions of the RSLES application. The functionality of the components and their interaction with one another, combined with the priority of design as identified by the user, drove the architectural design. We then constructed the system, placing the modules with the highest business value (but not necessarily the highest risk) at the top of the design list. Our priority was to develop the most desirable functionality first, then add additional functionality and 'nice to have' features as time permitted.

As modules were developed, users evaluated the system and revisions were made. RAD is an iterative process, so this method of development was normal and expected. The techniques of RAD were instrumental to our success, and provided a number of benefits. Based on our experience with RSLES, we support the view of Maner (1992) who states that this type of methodology tends to work better when:

- The application will run in a standalone vs. networked mode.
- Real-time performance is not critical.
- The system can be split into several independent, or loosely coupled, modules.
- The product is aimed at a highly specialized market.
- The required technology of the underlying components is more than a year old.

Had this application been required to be web-based or operate across a network, it would have been much more difficult to develop due to the additional levels of complexity associated with web-based application development. In its current configuration, only users with access to GAMS (specifically the CPLEX solver) will be able to use the program to its full benefit. The limited availability of GAMS among the individual services will diminish the usefulness of RSLES at organizational levels below the service recruiting commands.

Throughout the development process, data quality continually emerged as a limiting factor of the DSS application. Initial testing indicated that optimization of recruiting areas under certain conditions was infeasible, and some results provided an outcome that may not have been credible to the user. This identified the need to

incorporate the evaluation and assessment of source file data quality as a continuing effort throughout the development process.

C. SUMMARY

Developing a Decision Support System for the Services' relocation decision problem provided insight into new methods for improving the development methodology of a DSS. The Recruit Station Location Evaluation System (RSLES) is the result of using a RAD methodology to accelerate the development process, and subsequently reduce response time and costs. The system integrated three commercial software programs; GAMS as a model solver, MapInfo as a GIS mapping engine, and Access as the database management system. A user interface (UI), created in Visual Basic, served as an integration tool for retrieving data and passing information between these components.

The system architecture developed for the RSLES project consisted of an optimizer model, a data model, and an integrating application. The optimizer model was developed under separate research and constituted the basis for gathering the required data to evaluate the desirability of a recruit station location.

RSLES remains a prototypical application. Further enhancements will enable full functionality and improve maintainability. These enhancements include the incorporation of USAF and USMC recruiting stations, the development of a data warehouse with an automated data administrator, and the migration to a Web-based system. Each of these enhancements could be the subject of a future thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. RSLES SOURCE DATA FILE META-DATA.

This appendix contains the variables assigned to each zip code in the RSLES database and their definition.

Field Name	Data Type	Description	Source
Zip_Code	Text	Zip Code (Key Field)	Mapinfo
NStatZip	Number	Dummy Variable. 1 indicates Navy RS located within Zip Code.	Local
AstatZip	Number	Dummy Variable. 1 indicates Army RS located within Zip Code.	Local
JstatZip	Number	Dummy Variable. 1 indicates Joint RS located within Zip Code.	Local
Oprn_nv	Number	Number of full-time Navy recruiters stationed within Zip Code.	CNRC
Opra_ar	Number	Number of full-time Army recruiters stationed within Zip Code.	USAREC
Pop17	Number	Number of male, A-cell 17-21 year olds residing in Zip Code.	USAREC
Pop17old	Number	Number of 17-21 year olds residing within Zip Code.	USAREC
Year	Text	Fiscal Year of the data associated with this record.	Local
Quarter	Number	Fiscal Quarter of the data associated with this record.	Local
Ar_Battn	Text	Army Recruiting Battalion to which Zip Code is assigned.	USAREC
Ar_Co	Text	Army Recruiting Company to which Zip Code is assigned.	USAREC
Ar_Rs	Text	Army Recruiting Station to which Zip Code is assigned.	USAREC
Nv_Nrd	Text	Navy Recruiting District to which Zip Code is assigned.	CNRC
Nv_Nrz	Text	Navy Recruiting Zone to which Zip Code is assigned.	CNRC
Nv_Rs	Text	Navy Recruiting Station to which Zip Code is assigned.	CNRC
NoHS	Number	Number of high schools within Zip Code.	USAREC
HS1	Number	Dummy Variable. 1 indicates exactly one high school in Zip Code.	Local
HS2	Number	Dummy Variable. 1 indicates more than one high school in Zip	Local
Urate	Number	Unemployment rate. Rate applied to all Zip Codes within County.	Lewin Group
Area	Number	Area of Zip Code (Square Miles).	Lewin Group
Density	Number	Number of male A-cell 17-21 year olds per square mile within Zip	Local
Llat	Number	Latitude of centroid of Zip Code.	Mapinfo
Llong	Number	Longitude of centroid of Zip Code.	Mapinfo
HSSeniors	Number	Number of high school seniors residing within Zip Code.	USAREC
Fip	Text	Fip Code of Zip Code (Fip Code identifies State and County)	Lewin Group
METRO	Text	Metropolitan area of Zip Code. (Selected Zip Codes)	Census Bureau
INCOME	Number	Mean Household Income of Residents of Zip Code.	Lewin Group
PERCAPIN	Number	Per Capita Income of Residents of Zip Code.	Lewin Group

Field Name	Data Type	Description	Source
Adjust	Number	Variable used to determine estimated costs.	Lewin Group
Noreast	Number	Dummy Variable. 1 indicates Zip Code is in Northeastern Region.	Lewin Group
Midatl	Number	Dummy Variable. 1 indicates Zip Code is in Mid-Atlantic Region.	Lewin Group
Southatl	Number	Dummy Variable. 1 indicates Zip Code is in South Atlantic Region.	Lewin Group
Eastnc	Number	Dummy Variable. 1 indicates Zip Code is in Eastern North-Central Region.	Lewin Group
Eastsc	Number	Dummy Variable. 1 indicates Zip Code is in Eastern South-Central Region.	Lewin Group
Westnc	Number	Dummy Variable. 1 indicates Zip Code is in Western North-Central Region.	Lewin Group
Westsc	Number	Dummy Variable. 1 indicates Zip Code is in Western South-Central Region.	Lewin Group
Mountain	Number	Dummy Variable. 1 indicates Zip Code is in Mountain Region.	Lewin Group
Pacific	Number	Dummy Variable. 1 indicates Zip Code is in Pacific Region.	Lewin Group
Urban	Number	Dummy Variable. 1 indicates Zip Code is in an urban area.	Lewin Group
Suburban	Number	Dummy Variable. 1 indicates Zip Code is in a suburban area.	Lewin Group
Rural	Number	Dummy Variable. 1 indicates Zip Code is in a rural area.	Lewin Group
Costusa	Number	Estimated monthly base cost of operating an Army RS within Zip Code.	Local
Costusn	Number	Estimated monthly base cost of operating a Navy RS within Zip Code.	Local
CostJ2	Number	Estimated savings to each service operating in a Joint Station within Zip Code.	Local
CostRecSta	Number	Estimated costs in facilities for each recruiter within Zip Code.	Local
CostRecSal	Number	Estimated cost in salary for each recruiter within Zip Code.	Local
CostRec	Number	Total Cost for each recruiter within Zip Code.	Local

APPENDIX B. DESIGN PHASE ARTIFACTS

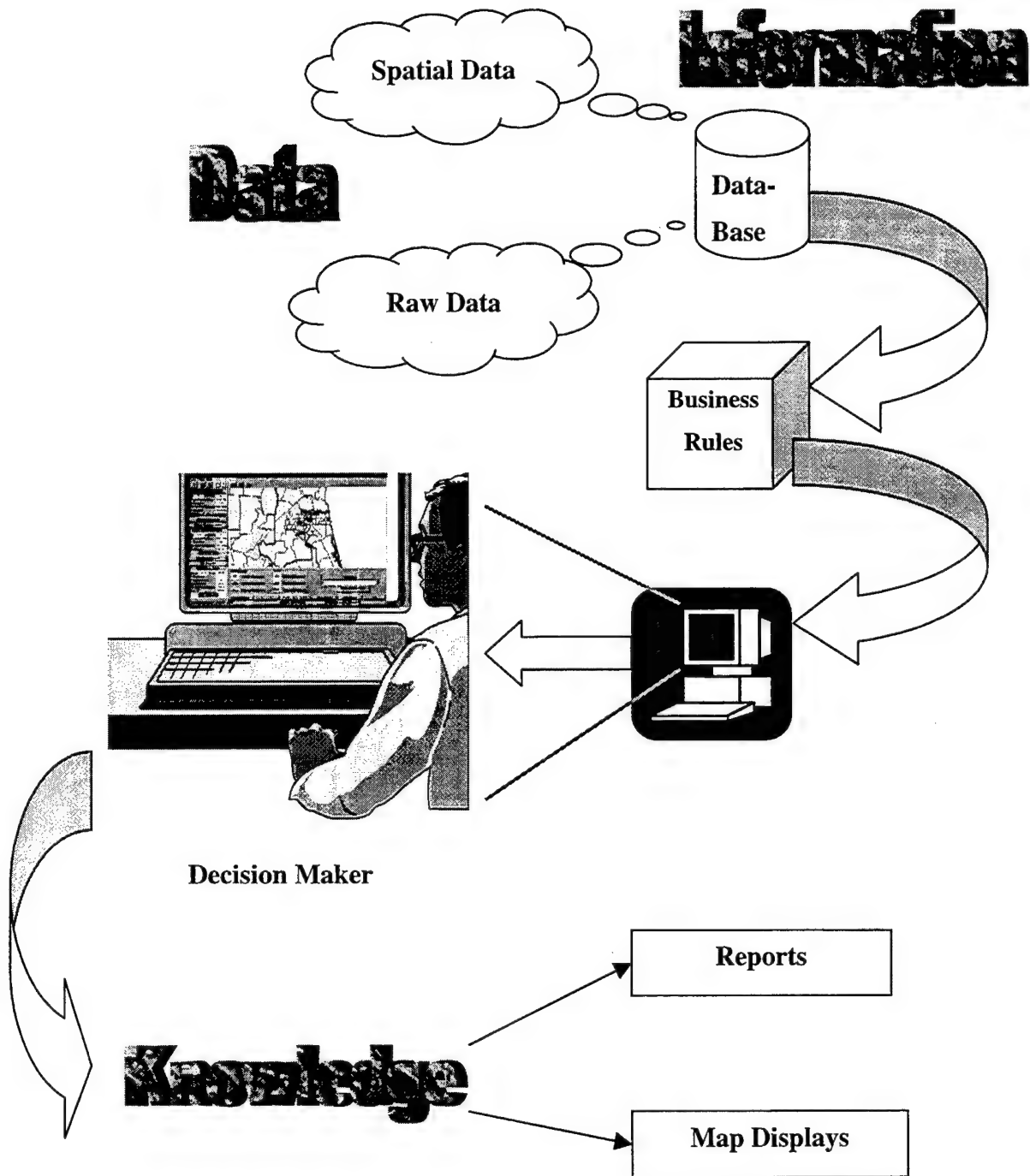
This appendix contains the artifacts that were documented upon completion of the Design Phase.

Index

1. Conceptual Overview Diagram.....	77
2. User Interface Initial Layout (Main View).....	79
3. User Interface Initial Layout (Map View).....	80
4. User Interface Final Layout (Main View).....	81
5. User Interface Final Layout (Map View).....	82

THIS PAGE LEFT INTENTIONALLY BLANK

Conceptual Overview Diagram



THIS PAGE LEFT INTENTIONALLY BLANK

User Interface Initial Layout.

The screenshot shows a window titled "Main Menu" with a dark background. At the top, it says "Toolbar goes here!". Below that is the title "Recruit Station Decision Support System". The main instruction is "Select a Service and a View". There are two columns of radio button options: "Service" with options "Air Force", "Army", "Marines", "Navy", and "Joint"; and "View" with options "District Level" and "Metropolitan Area". An "OK" button is centered below these columns. At the bottom right, there is a section labeled "RSDSS Controls" containing "Clear Form" and "Exit RSDSS" buttons. A horizontal bar with three empty rectangular boxes is located at the bottom left.

Main Menu

Toolbar goes here!

Recruit Station Decision Support System

Select a Service and a View

Service	View
<input type="radio"/> Air Force	<input type="radio"/> District Level
<input type="radio"/> Army	<input type="radio"/> Metropolitan Area
<input type="radio"/> Marines	
<input type="radio"/> Navy	
<input type="radio"/> Joint	

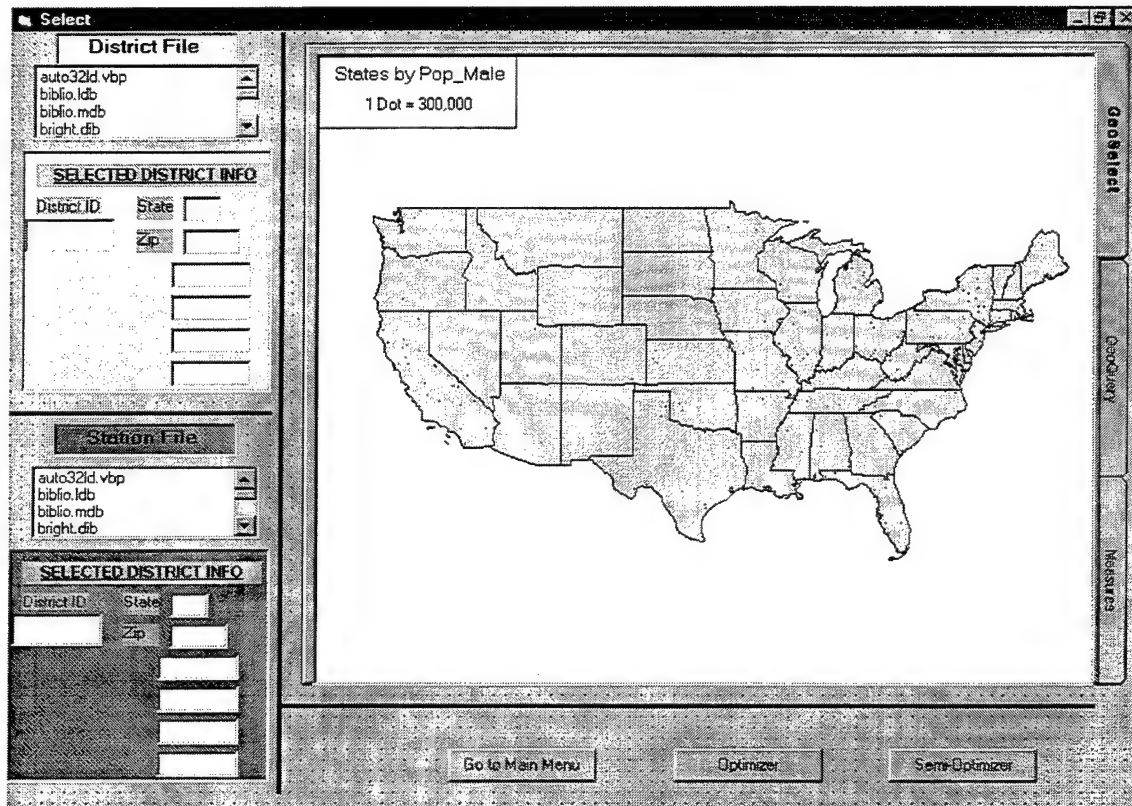
OK

RSDSS Controls

Clear Form Exit RSDSS

Main View

User Interface Initial Layout.



Map View

User Interface Final Layout.

Main Menu

Recruit Station Location Evaluation System

Select a Service, View, Unit, Year, and Quarter

Service	View	Unit	Year/Quarter
<input type="radio"/> Army	<input type="radio"/> Bn/District	Army Battalion <input type="text"/>	Fiscal Year <input type="text"/>
<input type="radio"/> Navy	<input type="radio"/> Company/ Zone	Army Company <input type="text"/>	Quarter <input type="text"/>
<input type="radio"/> Joint	<input type="radio"/> Metro	Metro <input type="text"/>	
		Navy District <input type="text"/>	
		Navy Zone <input type="text"/>	

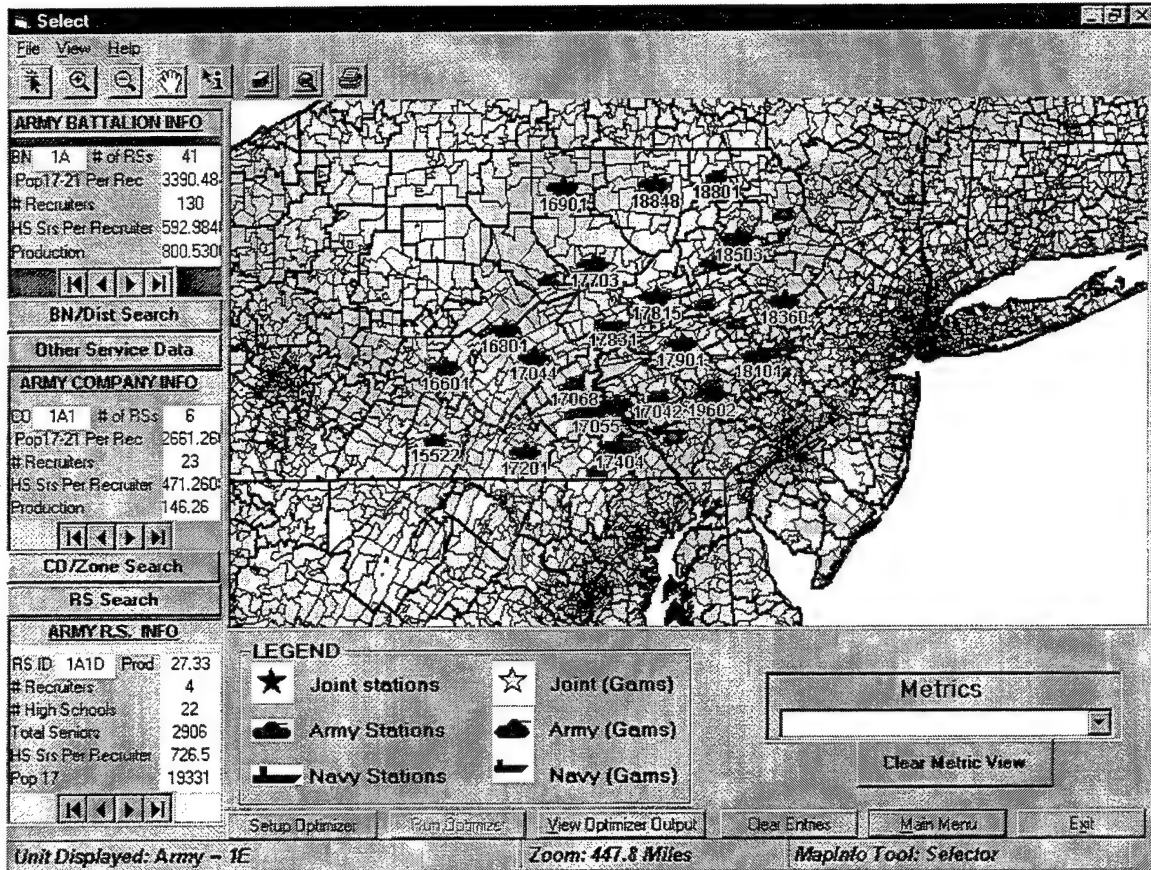
Continue

RSLES Controls

Clear Form Exit RSLES

Main View

User Interface Final Layout.



Map View

APPENDIX C. "MIN-COST" SOURCE/OUTPUT FILE LISTING.

This appendix contains examples of the source data files required for the optimizer and the output file created by the optimizer. All input files are tab delimited with the output file being comma delimited. The first seven files are input files and the last file (station.txt) is the output file. Input files are created in the 'Setup Optimizer' module and the output file is created in the 'Run Optimizer' module.

1. gamsin.std

Zip codes with stations and/or zip codes where the user desires to open/close/open or close a station(s). Attributes associated with each zip code are included in this file. This file contains the following fields: zip code, longitude, latitude, army recruiters, navy recruiters, cost per station, cost per service, population of 17-21 year olds, army status, and navy status. The army and navy status columns will have a 0, 1, or 2. The 1 indicates a station exists there or a station should be opened in that zip code. A zero (0) indicates that no station exists in that zip code or if a station does exist there, to close that station. A two (2) indicates a "candidate" station, that is, a station that is available to the min-cost model to be opened or closed or neither). NOTE: The data elements MUST line up under the headings (there is no heading for zip code).

Example:

	llong	llat	Arec	Nrec	lpop	Astatus	Nstatus
30034	-84.25	33.70	0	6	2916	0	1
30035	-84.21	33.73	4	0	1247	1	0
30080	-84.50	33.88	4	4	2110	1	1

2. gamsin.zpd Zip codes with attribute data for all zip codes in the unit selected (whether there is a station there or not).

Example:

	long	lat	pop	hs1	hs2	azip	nzip	jzip	area	density	income	urate	urban	suburb
30001	-84.612	33.812	2217	1	0	0	0	0	22.7	97.32	13880	0.04	1	0
30002	-84.263	33.774	268	0	1	0	0	0	1.80	148.88	18442	0.04	1	0
30021	-84.245	33.816	1309	1	0	0	0	0	3.12	419.55	14350	0.04	1	0

3. **gamsin.sti** Zip codes where stations reside in the unit selected (any service). The number of zip codes in this file will be considerably less than in the .zpi file and should equal the number of zip codes in the zzz.sti file.

Example:

30034
30035
30080
30083
30135

4. **gamsin.zpi** All zip codes in the domain of the unit selected. Generally, an Army company or Navy zone will have from 90 to 200 zip codes, Army battalions or Navy districts 200 to more than 500 zip codes, and for a metro area, it could be from 75 to over 300 zip codes.

Example:

30001
30002
30021
30027
30030
30032
30033
30034
30035

5. **gamsin.tgt** Production targets for Army or Navy. This is the sum of the gsma_army and gsma_navy data fields. The user can modify the target production number for their service only with the exception of the joint planner, who can modify both services production targets when optimizing a metro area.

Example:

Army 123
Navy 111

6. **gamsin.cst** The file that provides the GAMS optimizer with cost model values derived from the database. These values are listed only for zip codes that are listed in the gamsin.std file.

Example:

	coststa	costusa	costusn	costrec	costj2
30034	0	6458	6523	35000	-959
30035	0	6710	6775	35000	-959
30080	0	3765	3830	35000	-959

7. **zzz.zpi** Same as gamsin.zpi but with two columns of zip codes.

Example:

30001	30001
30002	30002
30021	30021
30027	30027
30030	30030
30032	30032
30033	30033
30034	30034
30035	30035

8. **zzz.sti** Same as gamsin.sti but with two columns of zip codes

Example:

30034	30034
30035	30035
30080	30080
30083	30083
30135	30135

9. **Stations.txt** The output file created during the optimization and contains a listing of all zip codes in the domain of the unit selected. This file provides uses a boolean value to indicate is a army, navy, or joint station resides in the zip code, the 'parent' station the zip code is associated with, the optimal number of recruiters that should be assigned to that station, and the estimated production from the zip code or station.

Example (header line does not 'wrap' around in the actual file):

zip_code	armysta	navysta	jointsta	armyrec	navyrec	armyzip	navyzip	armyprod	navyprod	armyrsprod	navyrsprod
"43316"	0,0,0,0,0	0	"45840"	"45840"	1.05	0.57	0.00	0.00			
"43323"	0,0,0,0,0	0	"44883"	"45840"	0.00	0.00	0.00	0.00			
"43330"	0,0,0,0,0	0	"45840"	"45840"	2.08	0.32	0.00	0.00			
"43351"	0,0,0,0,0	0	"45840"	"45840"	1.17	0.62	0.00	0.00			
"43359"	0,0,0,0,0	0	"45840"	"45840"	0.00	0.00	0.00	0.00			
"43402"	1,0,0,5,0	0	"43402"	"43615"	4.07	2.27	11.77	0.00			

APPENDIX D. RSLES SOURCE CODE LISTINGS

This appendix contains the detailed code listings of each Visual Basic module that comprise the RSLES application.

Index

Module 1. RSLES INITIAL USER INTERFACE.....	89
Module 2. RSLES MAIN USER INTERFACE.....	97
Module 3. SUBMAIN PROCEDURE.....	143
Module 4. RSLES PUBLIC DECLARATIONS.....	145
Module 5. MAPBASIC PUBLIC DECLARATIONS.....	147

THIS PAGE LEFT INTENTIONALLY BLANK

Module 1. RSLES INITIAL USER INTERFACE

Purpose: Contains all procedures and logic for controlling the initial U/I

Source Type: Visual Basic for Applications

Source File: MainMenu.frm

Code Listing:

Option Explicit
Option Base 1

Public iResponse As String
Public strSearchFor, strSQL, strSQL2, strSQL3 As String
Public iindex As Integer
Public RunOptCount As Integer 'Stores value of optimizer run
Public ComboBox1Filled, ComboBox2Filled As Boolean
Public ComboBox3Filled, ComboBox4Filled As Boolean

Private Sub cmdClear_Click()

'Clear the radio buttons

optArmy.Value = False

optNavy.Value = False

optJoint.Value = False

optDistrict.Value = False

optCoZone.Value = False

optMetro.Value = False

'Enable all combo boxes & set to blank

cboArmyBN.Enabled = True

cboArmyBN.Text = ""

cboArmyCo.Enabled = True

cboArmyCo.Text = ""

cboNavyDistrict.Enabled = True

cboNavyDistrict.Text = ""

cboNavyZone.Enabled = True

cboNavyZone.Text = ""

cboMetro.Enabled = True

cboMetro.Text = ""

cboYear.Text = ""

cboQtr.Text = ""

End Sub

Public Sub cmdExitBtn_Click()

On Error GoTo HandlecmdExitBtnError

'Close MapInfo files before killing. Otherwise error #75 results

MIObj.Do "close all interactive"

Kill (cMapPath & "Map*. *")

'close out objects and release all memory

Set MIObj = Nothing

```

End

cmdExitBtn_Exit:
Exit Sub

HandlecmdExitBtnError:
Dim stNewDatabaseName As String
Select Case Err.Number
Case 53
Resume Next
Case 91
Resume Next
Case Else
'MsgBox Err.Description, vbOKOnly + vbExclamation, "Unexpected Error"
Resume Next
End Select
End
End Sub

Private Sub cmdContinue_Click()
'we should redraw map
g_bRefreshMap = True

Dim criteria As String

'Check to see if a Service was selected
If optArmy.Value = False And optNavy.Value = False And optJoint.Value = False Then
Call DisplayMsg("Select a Service to continue", vbInformation, vbOKOnly, _
"Input Message")
optArmy.SetFocus 'Make the insertion point appear at the Rslesdb available choice
Exit Sub
Else 'service selected, check which one
If optArmy = True Then
If optDistrict.Value = False And optCoZone.Value = False Then
Call DisplayMsg("Select a View to continue", vbInformation, vbOKOnly, _
"Input Message")
optDistrict.SetFocus 'Make the insertion point appear at the first available choice
Exit Sub
Else
If optDistrict = True Then
UnitSelected = cboArmyBN.Text
Else
If optCoZone = True Then
UnitSelected = cboArmyCo.Text
End If
End If
End If
ElseIf optNavy.Value = True Then
If optDistrict.Value = False And optCoZone.Value = False Then
Call DisplayMsg("Select a View to continue", vbInformation, vbOKOnly, _
"Input Message")
optDistrict.SetFocus 'Make the insertion point appear at the first available choice
Exit Sub
Else
If optDistrict = True Then
UnitSelected = cboNavyDistrict.Text

```

```

Else
    If optCoZone = True Then
        UnitSelected = cboNavyZone.Text
    End If
End If
End If
ElseIf optJoint.Value = True Then
    UnitSelected = cboMetro.Text
End If
End If

'check if user selected valid year & quarter
If cboYear.Text = "" Or cboQtr.Text = "" Then
    strMsg = "You must first select a Year and Quarter"
    Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Input Message")
Exit Sub
Else
    strSQL = "SELECT Year, quarter FROM Rslesdb WHERE "
    strSQL2 = "Year like " & cboYear.Text & " and quarter = " & cboQtr.Text
    strSQL = strSQL + strSQL2

    datFormLoad.RecordSource = strSQL
    datFormLoad.Refresh

    With datFormLoad
        .Recordset.FindFirst "Year = " & cboYear.Text & " and Quarter = " & cboQtr.Text
        If datFormLoad.Recordset.NoMatch = True Then
            strMsg = "Rsles does not have data for the entered year/quarter." & vbCrLf & _
                "Please make another selection."
            Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "User Message")
            Exit Sub
        End If
    End With
End If

'Check if unit was actually selected
If UnitSelected = "" Then
    strMsg = "You did not select a unit."
    Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Input Message")
Exit Sub
Else
    strMsg = "Is " & UnitSelected & " your choice?"
    iResponse = MsgBox(strMsg, vbYesNo + vbQuestion, "Input Message")
    If iResponse = vbNo Then
        Exit Sub
    End If
End If

'convert metro names to database equivalent code
If frmMain.optJoint = True Then
    Select Case UnitSelected
        Case "Atlanta"
            Let UnitSelected = "ATL"
        Case "Boston"
            Let UnitSelected = "BOS"
    End Select
End If

```

```

Case "Chicago"
    Let UnitSelected = "CHI"
Case "Dallas"
    Let UnitSelected = "DAL"
Case "Denver"
    Let UnitSelected = "DEN"
Case "Houston"
    Let UnitSelected = "HOU"
Case "Jacksonville"
    Let UnitSelected = "JAX"
Case "Los Angeles"
    Let UnitSelected = "LA"
Case "New York City"
    Let UnitSelected = "NYC"
Case "San Diego"
    Let UnitSelected = "SD"
Case "San Francisco"
    Let UnitSelected = "SF"
End Select
End If

Call DisplayMsg("The RSLES main form may take awhile to load.", _
    vbInformation, vbOKOnly, "User Message")

'set variables 'unit' and 'RName' (and RName2 for Joint)
If frmMain.optArmy And frmMain.optDistrict.Value = True Then
    unit = "ar_battn"
    RName = "Ar_Rs"
    RName2 = "Nv_Rs"
    Service = "Army"
ElseIf frmMain.optArmy And frmMain.optCoZone.Value = True Then
    unit = "ar_co"
    RName = "Ar_Rs"
    RName2 = "Nv_Rs"
    Service = "Army"
ElseIf frmMain.optNavy.Value = True And frmMain.optDistrict.Value = True Then
    unit = "nv_nrd"
    RName = "Nv_Rs"
    RName2 = "Ar_Rs"
    Service = "Navy"
ElseIf frmMain.optNavy.Value = True And frmMain.optCoZone.Value = True Then
    unit = "nv_nrz"
    RName = "Nv_Rs"
    RName2 = "Ar_Rs"
    Service = "Navy"
ElseIf frmMain.optJoint = True And frmMain.optMetro = True Then
    unit = "metro"
    RName = "Ar_Rs"
    RName2 = "Nv_Rs"
    Service = "Joint"
End If

Call DisplayPanelMsg("Unit Displayed: " & Service & " -- " & UnitSelected)

Screen.MousePointer = vbHourglass

```

```

frmSplash.Show vbModeless 'Display the splash form
frmSplash.Refresh 'force the splash screen to repaint itself
frmMain.Hide 'hide the intro form
Load frmSelect 'Load the main form
frmSelect.Show 'show the main form
Screen.MousePointer = vbDefault

```

```
End Sub
```

```

Private Sub Form_Load()
'Establish DEFAULT position for Main Menu
Me.Top = 0
Me.Left = 0

```

```
On Error GoTo HandleFormLoadError
```

```

'delete files created for FormLoad
Kill (cMapPath & "Map*.")

```

```

'delete files created for Map display
Kill (cMapPath & "Nstation.*")
Kill (cMapPath & "Astation.*")
Kill (cMapPath & "station.map")
Kill (cMapPath & "station.tab")
Kill (cMapPath & "station.id")
Kill (cMapPath & "station.ind")

```

```

'Fill combo boxes
Call FillComboBoxes

```

```

Form_Exit:
Exit Sub

```

```
HandleFormLoadError:
```

```

Select Case Err.Number
Case 53, 75 'file not found, path error
Resume Next
Case 70
Resume Next
Case Else
MsgBox "Error Number:" + Str(Err.Number) + _
"; Description: " + Err.Description + _
".", vbInformation, _
"The Friendly Error Handler"
Resume Next
End Select

```

```
End Sub
```

```

Private Sub optArmy_Click()
Let optArmy.Value = True
End Sub

```

```

Private Sub optCoZone_Click()
cboMetro.Enabled = False

```

```

If optArmy.Value = True Then
    cboArmyCo.Enabled = True
    cboArmyBN.Enabled = False
    cboNavyDistrict.Enabled = False
    cboNavyZone.Enabled = False
ElseIf optNavy.Value = True Then
    cboNavyZone.Enabled = True
    cboArmyCo.Enabled = False
    cboArmyBN.Enabled = False
    cboNavyDistrict.Enabled = False
Else
    strMsg = "First select a Service."
    Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Input Message")
End If
End Sub

```

```

Private Sub optDistrict_Click()
    cboMetro.Enabled = False
    If optArmy.Value = True Then
        cboArmyBN.Enabled = True
        cboArmyCo.Enabled = False
        cboNavyDistrict.Enabled = False
        cboNavyZone.Enabled = False
    ElseIf optNavy.Value = True Then
        cboNavyDistrict.Enabled = True
        cboArmyCo.Enabled = False
        cboArmyBN.Enabled = False
        cboNavyZone.Enabled = False
    Else
        strMsg = "First select a Service."
        Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Input Message")
    End If
End Sub

```

```

Private Sub optJoint_Click()

    Let optJoint.Value = True

End Sub

```

```

Private Sub optMetro_Click()
    If optJoint.Value = True Then
        cboMetro.Enabled = True
        cboArmyCo.Enabled = False
        cboArmyBN.Enabled = False
        cboNavyDistrict.Enabled = False
        cboNavyZone.Enabled = False
    Else
        strMsg = "First select a Service."
        Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Input Message")
    End If
End Sub

```

```

Private Sub optNavy_Click()

```

```
Let optNavy.Value = True
End Sub
```

```
Private Function FillComboBoxes()
```

```
    cboArmyBN.Clear      'Clear Selection 1 list box (Bn/Dist)
    cboArmyCo.Clear      'Clear Selection 2 list box (Co/Zone)
    cboNavyDistrict.Clear
    cboNavyZone.Clear
    cboYear.Clear
```

```
    ComboBox1Filled = False 'Army Bn box
    ComboBox2Filled = False 'Army Co box
    ComboBox3Filled = False 'Navy District box
    ComboBox4Filled = False 'Navy Zone box
```

```
    'Call DisplayPanelMsg("Optimize: Select a zone/company")
    Screen.MousePointer = vbHourglass
```

```
    'Fill Army Bn ComboBox
    strSQL = "SELECT DISTINCT ar_battn FROM Rslesdb"
    datFormLoad.RecordSource = strSQL
    datFormLoad.Refresh
    'populate army combo box 1 (BNs)
    With datFormLoad
        Do Until .Recordset.EOF
            cboArmyBN.AddItem .Recordset!ar_battn
            .Recordset.MoveNext
        Loop
    End With
    datFormLoad.Recordset.MoveFirst
    ComboBox1Filled = True
```

```
    'Fill Army Company list box
    strSQL = "SELECT DISTINCT Ar_Co FROM Rslesdb"
    datFormLoad.RecordSource = strSQL
    datFormLoad.Refresh
```

```
    'populate army combo box 2 (Companies)
    With datFormLoad
        Do Until .Recordset.EOF
            cboArmyCo.AddItem .Recordset!Ar_Co
            .Recordset.MoveNext
        Loop
    End With
    datFormLoad.Recordset.MoveFirst
    ComboBox2Filled = True
```

```
    'Fill Navy District ComboBox
    strSQL = "SELECT DISTINCT nv_nrd FROM Rslesdb"
    datFormLoad.RecordSource = strSQL
    datFormLoad.Refresh
    'populate combo box 3 (Navy Districts)
    With datFormLoad
        Do Until .Recordset.EOF
            cboNavyDistrict.AddItem .Recordset!Nv_Nrd
```

```

        .Recordset.MoveNext
    Loop
End With
datFormLoad.Recordset.MoveFirst
ComboBox3Filled = True

'Populate Navy combo box 2 (Zones)
strSQL = "SELECT DISTINCT Nv_Nrz FROM Rslesdb"
datFormLoad.RecordSource = strSQL
datFormLoad.Refresh
'populate combo box 4 (Navy Zones)
With datFormLoad
    Do Until .Recordset.EOF
        cboNavyZone.AddItem .Recordset!Nv_Nrz
        .Recordset.MoveNext
    Loop
End With
datFormLoad.Recordset.MoveFirst
ComboBox4Filled = True

'Populate year combo box
strSQL = "SELECT DISTINCT Year FROM Rslesdb"
datFormLoad.RecordSource = strSQL
datFormLoad.Refresh
'populate year combo box
With datFormLoad
    Do Until .Recordset.EOF
        cboYear.AddItem .Recordset!Year
        .Recordset.MoveNext
    Loop
End With
datFormLoad.Recordset.MoveFirst

'Populate quarter combo box
strSQL = "SELECT DISTINCT Quarter FROM Rslesdb"
datFormLoad.RecordSource = strSQL
datFormLoad.Refresh
'populate quarter combo box
With datFormLoad
    Do Until .Recordset.EOF
        cboQtr.AddItem .Recordset!Quarter
        .Recordset.MoveNext
    Loop
End With
datFormLoad.Recordset.MoveFirst

Screen.MousePointer = vbDefault

End Function

```

Module 2. RSLES MAIN USER INTERFACE

Purpose: Contains all procedures and logic for controlling the main U/I

Source Type: Visual Basic for Applications

Source File: Select.frm

Code Listing:

Option Explicit
Option Base 1

```
Private Declare Function GetExitCodeProcess Lib "kernel32" _
    (ByVal hProcess As Long, lpExitCode As Long) As Long
Private Declare Function OpenProcess Lib "kernel32" _
    (ByVal dwDesiredAccess As Long, _
    ByVal bInheritHandle As Long, _
    ByVal dwProcessId As Long) As Long
Private Declare Function WaitForSingleObject Lib "kernel32" _
    (ByVal hHandle As Long, ByVal dwMilliseconds As Long) As Long
Private Declare Function CloseHandle Lib "kernel32" _
    (ByVal hObject As Long) As Long
Private Const INFINITE = &HFFFF

!*****
!*
!*  GeoCode
!*
!*  This routine loops through the table processing each row.
!*
!*****

Sub GeoCode(sAddressTable As String, sSymbol As String)
'sAddressTable is the table we want to geocode
'sSymbol is a "A","B","J","N","R",or "Y" depending on what symbol you wish to use

Dim iResult As Integer

'On Error GoTo HandleGeocodeError

MIObj.Do ("aRowid = " & sAddressTable & ".rowid")
'assuming the zip column in your table is called "ZIP_CODE"
MIObj.Do ("aAddress = " & sAddressTable & ".ZIP_CODE")
'create a map for the new table
MIObj.Do (" Create Map For " & sAddressTable & " CoordSys Earth Projection 1, 0")
'tell MapInfo what table we're searching against
MIObj.Do ("Find Using US_ZIPS("""ZIP_CODE""")")
'get the first record from the table
MIObj.Do ("Fetch First from " & sAddressTable)      ' cursor to first record
'loop through until we hit the end of the table
Do While MIObj.eval("EOT(" & sAddressTable & ")") = "F"
```

```

MIObj.Do ("iCounter = aRowid ")
MIObj.Do ("Find aAddress")
'get the results of the find (a postive number means it found the zip)
iResult = MIObj.eval("CommandInfo(3)")

If iResult < 0 Then
'Record not geocoded
Else
'based on the symbol, set the point's style
Select Case sSymbol
'the parameters for the MakeFontSymbol are:
'makeFontSymbol(symbol#, color, size, font name, fontstyle*, rotation angle)
' *fontstyle - 1 bold, 256 a white halo, 257 (1+256) = bold halo
'
'         16 is a black outline

    Case "J" 'purple star
        MIObj.Do ("symPoint = makeFontSymbol(36, 8388736, 12, ""MapInfo Symbols"", 16, 0)")
'joint
    Case "N" 'blue ship
        MIObj.Do ("symPoint = makeFontSymbol(94, 255, 12, ""MapInfo Transportation"", 16, 0)")
'navy
    Case "A" 'army tank
        MIObj.Do ("symPoint = makeFontSymbol(92, 32768, 12, ""MapInfo Transportation"", 16,
0)") 'army
    Case "B" 'brown tank (army GAMS stations)
        MIObj.Do ("symPoint = makeFontSymbol(92, 9464832, 12, ""MapInfo Transportation"", 16,
0)") 'army
    Case "R" 'red ship (navy stations GAMS)
        MIObj.Do ("symPoint = makeFontSymbol(94, 16711680, 12, ""MapInfo Transportation"",
16, 0)") 'navy
    Case "Y" 'yellow star (joint stations GAMS)
        MIObj.Do ("symPoint = makeFontSymbol(36, 16776960, 12, ""MapInfo Symbols"", 16, 0)")
'unit stations (load)
End Select
'need diff color here
'get the coords of where we found the zip
MIObj.Do (" x = CommandInfo(1)") 'X coord of find
MIObj.Do ("y = CommandInfo(2)") 'Y coord of find
'set the style for the new symbol
MIObj.Do ("Set Style Symbol symPoint")
'update the table to include the new symbol
MIObj.Do ("Update " & sAddressTable & " Set obj = CreatePoint(x, y) Where rowid = iCounter")
End If
'get next record
MIObj.Do ("Fetch Next from " & sAddressTable)

Loop

MIObj.Do ("Commit table " & sAddressTable)

'Geocode_Exit:
' Exit Sub

'HandleGeocodeError:

' Select Case Err.Number
' Case Else

```

```
' Resume
' End Select
```

```
End Sub
```

```
Private Sub cboMetrics_Click()
```

```
'Displays metrics by Service station or in the case of Joint, by zip code
```

```
On Error GoTo HandleMetrics_Error
```

```
Call DisplayPanelMsg(ThematicMapMSG)
```

```
If Service = "Army" Or Service = "Navy" Then
```

```
    If cboMetrics.ListIndex = 0 And Service = "Army" Then '% Market Share
```

```
        strSQL = "SELECT " & RSname & ", (Sum(gsma_army)/Sum(gsma_dod)) "
```

```
        strSQL2 = "AS MktShare From Rslesdb WHERE Year = " & frmMain.cboYear & " and "
```

```
        strSQL3 = "quarter = " & frmMain.cboQtr.Text & " and " & unit & " = " & UnitSelected & "
```

```
    Group by " & RSname & ""
```

```
    ElseIf cboMetrics.ListIndex = 0 And Service = "Navy" Then
```

```
        strSQL = "SELECT " & RSname & ", (Sum(gsma_navy)/Sum(gsma_dod)) "
```

```
        strSQL2 = "AS MktShare From Rslesdb WHERE Year = " & frmMain.cboYear & " and "
```

```
        strSQL3 = "quarter = " & frmMain.cboQtr.Text & " and " & unit & " = " & UnitSelected & "
```

```
    Group by " & RSname & ""
```

```
    ElseIf cboMetrics.ListIndex = 1 Then '# High Schools per RS
```

```
        strSQL = "SELECT " & RSname & ", SUM(NoHs) As TotHS FROM Rslesdb "
```

```
        strSQL2 = "WHERE Year = " & frmMain.cboYear & " and "
```

```
        strSQL3 = "quarter = " & frmMain.cboQtr.Text & " and " & unit & " = " & UnitSelected & "
```

```
    Group by " & RSname & ""
```

```
    ElseIf cboMetrics.ListIndex = 2 Then '# 17-21 year olds per RS
```

```
        strSQL = "SELECT " & RSname & ", SUM(Pop17) As Tot17 FROM Rslesdb "
```

```
        strSQL2 = "WHERE Year = " & frmMain.cboYear & " and quarter "
```

```
        strSQL3 = "= " & frmMain.cboQtr.Text & " and " & unit & " = " & UnitSelected & " GROUP BY "
```

```
    & RSname & ""
```

```
End If
```

```
datTempQuery.RecordSource = strSQL + strSQL2 + strSQL3
```

```
datTempQuery.Refresh
```

```
cMapFileName = "MapMetrics"
```

```
Call QueryConvert(datTempQuery.Recordset, cMapFileName) 'Write files required
```

```
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
```

```
    "MapMetrics.tab" & Chr$(34) & " Interactive")
```

```
'Get the handle to the front window
```

```
iMapInfoWinID = MIObj.eval("frontwindow()")
```

```
'Do thematic mapping
```

```
If Service = "Army" And cboMetrics.ListIndex = 0 Then
```

```
    'The first and second Ar_Rs are the table 'Ar_Rs', the last AR_RS is the field ar_rs from the database
```

```
    MIObj.Do ("Add Column ""Ar_Rs""(MktShare float) from mapMetrics set to MktShare where ""AR_RS"" = ""AR_RS""")
```

```
    MIObj.Do ("shade window frontwindow() ""AR_RS"" with MktShare ranges apply all use color  
Brush (2,16711680,16777215) 0:0.10 Brush (2,16711680,16777215) Pen (1,2,0) ,0.10:0.20  
Brush (2,12591104,16777215) Pen (1,2,0) ,0.20:0.30 Brush (2,8404992,16777215) Pen (1,2,0)  
,0.30:0.40 Brush (2,4218880,16777215) Pen (1,2,0) ,0.40:0.99 Brush (2,32768,16777215) Pen  
(1,2,0) default Brush (2,16777215,16777215) Pen (1,2,0)")
```

```

ElseIf frmMain.optNavy = True And cboMetrics.ListIndex = 0 Then
    MIObj.Do ("Add Column ""Nv_Rs""(MktShare float) from mapMetrics set to MktShare where
""NV_RS"" = ""NV_RS""")
    MIObj.Do ("shade window frontwindow() ""NV_RS"" with MktShare ranges apply all use color
Brush (2,16711680,16777215) 0:0.10 Brush (2,16711680,16777215) Pen (1,2,0) ,0.10:0.20
Brush (2,12591104,16777215) Pen (1,2,0) ,0.20:0.30 Brush (2,8404992,16777215) Pen (1,2,0)
,0.30:0.40 Brush (2,4218880,16777215) Pen (1,2,0) ,0.40:0.99 Brush (2,32768,16777215) Pen
(1,2,0) default Brush (2,16777215,16777215) Pen (1,2,0)")
    ElseIf Service = "Army" And cboMetrics.ListIndex = 1 Then
        MIObj.Do ("Add Column ""Ar_Rs""(TotHS integer) from mapMetrics set to TotHS where
""AR_RS"" = ""AR_RS""")
        MIObj.Do ("shade window frontwindow() ""AR_RS"" with totHS ranges apply all use color
Brush (2,16711680,16777215) 0:5 Brush (2,16711680,16777215) Pen (1,2,0) ,5:10 Brush
(2,12591104,16777215) Pen (1,2,0) ,10:18 Brush (2,8404992,16777215) Pen (1,2,0) ,18:30
Brush (2,4218880,16777215) Pen (1,2,0) ,30:100 Brush (2,32768,16777215) Pen (1,2,0) default
Brush (2,16777215,16777215) Pen (1,2,0)")
        ElseIf frmMain.optNavy = True And cboMetrics.ListIndex = 1 Then
            MIObj.Do ("Add Column ""Nv_Rs""(TotHS integer) from mapMetrics set to TotHS where
""NV_RS"" = ""NV_RS""")
            MIObj.Do ("shade window frontwindow() ""NV_RS"" with totHS ranges apply all use color
Brush (2,16711680,16777215) 0:5 Brush (2,16711680,16777215) Pen (1,2,0) ,5:10 Brush
(2,12591104,16777215) Pen (1,2,0) ,10:18 Brush (2,8404992,16777215) Pen (1,2,0) ,18:30
Brush (2,4218880,16777215) Pen (1,2,0) ,30:100 Brush (2,32768,16777215) Pen (1,2,0) default
Brush (2,16777215,16777215) Pen (1,2,0)")
            ElseIf Service = "Army" And cboMetrics.ListIndex = 2 Then
                MIObj.Do ("Add Column ""Ar_Rs""(Tot17 integer) from mapMetrics set to Tot17 where
""AR_RS"" = ""AR_RS""")
                MIObj.Do ("shade window frontwindow() ""AR_RS"" with tot17 ranges apply all use color Brush
(2,16711680,16777215) 0:5000 Brush (2,16711680,16777215) Pen (1,2,0) ,5000:10000 Brush
(2,12591104,16777215) Pen (1,2,0) ,10000:20000 Brush (2,8404992,16777215) Pen (1,2,0)
,20000:30000 Brush (2,4218880,16777215) Pen (1,2,0) ,30000:75000 Brush
(2,32768,16777215) Pen (1,2,0) default Brush (2,16777215,16777215) Pen (1,2,0)")
                ElseIf frmMain.optNavy = True And cboMetrics.ListIndex = 2 Then
                    MIObj.Do ("Add Column ""Nv_Rs""(Tot17 integer) from mapMetrics set to Tot17 where
""NV_RS"" = ""NV_RS""")
                    MIObj.Do ("shade window frontwindow() ""NV_RS"" with tot17 ranges apply all use color Brush
(2,16711680,16777215) 0:5000 Brush (2,16711680,16777215) Pen (1,2,0) ,5000:10000 Brush
(2,12591104,16777215) Pen (1,2,0) ,10000:20000 Brush (2,8404992,16777215) Pen (1,2,0)
,20000:30000 Brush (2,4218880,16777215) Pen (1,2,0) ,30000:75000 Brush
(2,32768,16777215) Pen (1,2,0) default Brush (2,16777215,16777215) Pen (1,2,0)")
                    End If

'Create the legend for the map
MIObj.Do "set next document parent windowinfo(frontwindow(), 12) style 1"
MIObj.Do "Create legend from window frontwindow()"

```

Else 'Joint metrics

```

If cboMetrics.ListIndex = 0 And Service = "Joint" Then 'Accessions (all Services)
    strSQL = "SELECT zip_code, Gsma_dod From Rslesdb "
    strSQL2 = "WHERE Year = " & frmMain.cboYear & " and "
    strSQL3 = "quarter = " & frmMain.cboQtr.Text & " and " & unit & " = " & UnitSelected & ""
ElseIf cboMetrics.ListIndex = 1 Then '# High Schools per RS
    strSQL = "SELECT zip_code, NoHs FROM Rslesdb "
    strSQL2 = "WHERE Year = " & frmMain.cboYear & " and "
    strSQL3 = "quarter = " & frmMain.cboQtr.Text & " and " & unit & " = " & UnitSelected & ""
ElseIf cboMetrics.ListIndex = 2 Then '# 17-21 year olds per RS

```

```

strSQL = "SELECT zip_code, Pop17 FROM Rslesdb "
strSQL2 = "WHERE Year = " & frmMain.cboYear & " and quarter "
strSQL3 = "= " & frmMain.cboQtr.Text & " and " & unit & " = " & UnitSelected & ""
End If

```

```

datTempQuery.RecordSource = strSQL + strSQL2 + strSQL3
datTempQuery.Refresh
cMapFileName = "MapMetrics"

```

```

Call QueryConvert(datTempQuery.Recordset, cMapFileName) 'Write files required
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"MapMetrics.tab" & Chr$(34) & " Interactive")

```

```

'Do thematic mapping

```

```

If cboMetrics.ListIndex = 0 Then

```

```

    MIObj.Do ("Select mapmetrics.zip_code, mapmetrics.Gsma_Dod, Uszipbdy.obj from
mapmetrics, uszipbdy where mapmetrics.zip_code = uszipbdy.zip into mapmetrics1")

```

```

    MIObj.Do ("Add Map Layer mapmetrics1")

```

```

    MIObj.Do ("shade window frontwindow() ""mapmetrics1"" with gsma_dod ranges apply all use
color Brush (2,16711680,16777215) 0:5 Brush (2,16711680,16777215) Pen (1,2,0) ,5:10 Brush
(2,12591104,16777215) Pen (1,2,0) ,10:15 Brush (2,8404992,16777215) Pen (1,2,0) ,15:20
Brush (2,4218880,16777215) Pen (1,2,0) ,20:100 Brush (2,32768,16777215) Pen (1,2,0) default
Brush (2,16777215,16777215) Pen (1,2,0)")

```

```

    Elseif cboMetrics.ListIndex = 1 Then

```

```

        MIObj.Do ("Select mapmetrics.Zip_code, mapmetrics.NoHs, Uszipbdy.obj from mapmetrics,
uszipbdy where mapmetrics.zip_code = uszipbdy.zip into mapmetrics1")

```

```

        MIObj.Do ("Add Map Layer mapmetrics1")

```

```

        MIObj.Do ("shade window frontwindow() ""Mapmetrics1"" with NoHS ranges apply all use color
Brush (2,16711680,16777215) 0:0 Brush (2,16711680,16777215) Pen (1,2,0) ,1:1 Brush
(2,12591104,16777215) Pen (1,2,0) ,2:2 Brush (2,8404992,16777215) Pen (1,2,0) ,3:3 Brush
(2,4218880,16777215) Pen (1,2,0) ,4:10 Brush (2,32768,16777215) Pen (1,2,0) default Brush
(2,16777215,16777215) Pen (1,2,0)")

```

```

        Elseif cboMetrics.ListIndex = 2 Then

```

```

            MIObj.Do ("Select mapmetrics.zip_code, mapmetrics.Pop17, Uszipbdy.obj from mapmetrics,
uszipbdy where mapmetrics.zip_code = uszipbdy.zip into mapmetrics1")

```

```

            MIObj.Do ("Add Map Layer mapmetrics1")

```

```

            MIObj.Do ("shade window frontwindow() ""mapmetrics1"" with pop17 ranges apply all use color
Brush (2,16711680,16777215) 0:500 Brush (2,16711680,16777215) Pen (1,2,0) ,500:1000
Brush (2,12591104,16777215) Pen (1,2,0) ,1000:1500 Brush (2,8404992,16777215) Pen (1,2,0)
,1500:2000 Brush (2,4218880,16777215) Pen (1,2,0) ,2000:15000 Brush (2,32768,16777215)
Pen (1,2,0) default Brush (2,16777215,16777215) Pen (1,2,0)")

```

```

        End If

```

```

'Get the handle to the front window

```

```

iMapInfoWinID = MIObj.eval("frontwindow()")

```

```

'Create the legend for the map

```

```

MIObj.Do "set next document parent windowinfo(frontwindow(), 12) style 1"

```

```

MIObj.Do "Create legend from window frontwindow()"

```

```

End If

```

```

DisplayPanelMsg ("Select 'CLEAR METRIC VIEW' to continue.")

```

```

cboMetrics_Exit:

```

```

Exit Sub

```

HandleMetrics_Error:

```
Select Case Err.Number
  Case 70
    MIObj.Do ("close Table MapMetrics interactive")
    Resume
  Case 53, 75 'file not found
    Resume Next
  Case Else
    MsgBox "Error Number: " + Str(Err.Number) + _
      "; Description: " + Err.Description + _
      ". ", vbInformation, _
      "The Friendly Error Handler"
    Resume Next
End Select
```

End Sub

Private Sub cmdClear_Click()

```
'Loop to clear control arrays for opening and closing
For iindex = 1 To 10
  txtAOpenCloseRS(iindex) = ""
  txtAopenRS(iindex) = ""
  txtAcloseRS(iindex) = ""
Next iindex
End Sub
```

Private Sub cmdCoZoneSearch_Click()

Call DisplayPanelMsg("Searching for CO/Zone")

' Determine service to search for

Let strSearchFor2 = LCase(InputBox("Enter Service to search (Army, Navy)"))

If strSearchFor2 <> "army" And strSearchFor2 <> "navy" Then

Call DisplayMsg("Invalid service selection.", vbInformation, vbOKOnly, "Input Message")

Exit Sub

End If

' Search for the CO or Zone specified by the user

Let strSearchFor = LCase(InputBox("Enter RS ID of the Recruiting Station ID: "))

If Len(strSearchFor) > 0 Then

If strSearchFor2 = "army" Then

pctRSArmy.Visible = True

pctArmyCO.Visible = True

pctArmyBN.Visible = True

pctRSNavy.Visible = False

pctNavyZone.Visible = False

pctNavyDistrict.Visible = False

datArmyCO.Recordset.FindFirst "Ar_Co = " & strSearchFor & ""

If datArmyCO.Recordset.NoMatch Then

MsgBox "No Army Company match found", vbExclamation, "Find a Company"

End If

Elseif strSearchFor2 = "navy" Then

pctRSNavy.Visible = True

pctNavyZone.Visible = True

```

pctNavyDistrict.Visible = True
pctArmyBN.Visible = False
pctArmyCO.Visible = False
pctRSArmy.Visible = False
datNavyZone.Recordset.FindFirst "Nv_nrz = " & strSearchFor & ""
If datNavyZone.Recordset.NoMatch Then
    MsgBox "No Navy Zone match found", vbExclamation, "Find a Zone"
End If
End If
Else
    Call DisplayMsg("Must enter a CO or Zone ID.", vbInformation, vbOKOnly, "Input Message")
End If

```

End Sub

```

Private Sub cmdDistSearch_Click()
    Call DisplayPanelMsg("Searching for BN/District")
    ' Determine service to search for
    Let strSearchFor2 = LCase(InputBox("Enter Service to search (Army, Navy)"))
    If strSearchFor2 <> "army" And strSearchFor2 <> "navy" Then
        Call DisplayMsg("Invalid service selection.", vbInformation, vbOKOnly, "Input Message")
        Exit Sub
    End If
    ' Search for the RS ID specified by the user
    Let strSearchFor = LCase(InputBox("Enter the BN/District ID to search for: "))

    If Len(strSearchFor) > 0 Then

        If strSearchFor2 = "army" Then
            'set correct data boxes to view
            pctArmyBN.Visible = True
            pctArmyCO.Visible = True
            pctRSArmy.Visible = True
            pctRSNavy.Visible = False
            pctNavyDistrict.Visible = False
            pctNavyZone.Visible = False
            vBookMark = datArmyBn.Recordset.Bookmark
            datArmyBn.Recordset.FindFirst "Ar_Battn = " & strSearchFor & ""
            If datArmyBn.Recordset.NoMatch = True Then
                MsgBox "No match found", vbExclamation, "Find a BN"
                datArmyBn.Recordset.Bookmark = vBookMark
            End If
        ElseIf strSearchFor2 = "navy" Then
            pctRSNavy.Visible = True
            pctNavyDistrict.Visible = True
            pctNavyZone.Visible = True
            pctArmyBN.Visible = False
            pctArmyCO.Visible = False
            pctRSArmy.Visible = False
            vBookMark = datNavyDistrict.Recordset.Bookmark
            datNavyDistrict.Recordset.FindFirst "Nv_nrd = " & strSearchFor & ""
            If datNavyDistrict.Recordset.NoMatch = True Then
                MsgBox "No match found", vbExclamation, "Find a Distirt"
                datNavyDistrict.Recordset.Bookmark = vBookMark
            End If
        End If
    End If

```

```

Else
    Call DisplayMsg("Must enter a BN/District ID.", vbInformation, vbOKOnly, "Input Message")
End If

```

```

End Sub

```

```

Private Sub cmdDoneGAMS_Click()
'Delete files that were created in MapInfo

```

```

On Error GoTo HandleDoneGamsError

```

```

MIObj.Do "close all interactive"
Kill (cMapPath & "Station.*")      'All Station files
Kill (cMapPath & "Astation.*")     'All Astation files
Kill (cMapPath & "Nstation.*")     'All Nstation files

```

```

'set screen view
cmdGoMain.Enabled = True
cmdSetupOptimizer.Enabled = True
cmdResetMetrics.Visible = True
pctMeasure.Visible = True
pctGAMSoutput.Visible = False
pctAsIsData.Visible = False
frmLegend.Visible = True
cmdDisplayGAMS.Enabled = False
cmdDoneGAMS.Visible = False
OpenCloseTabs.Visible = False
cmdClear.Enabled = False

```

```

'clear recordsets
Set datOptimize.Recordset = Nothing
Set datTempQuery.Recordset = Nothing
Set datTempQuery2.Recordset = Nothing
Set datTempQuery3.Recordset = Nothing
Set datValidateZip.Recordset = Nothing

```

```

'Clear labels
lblJointStat = ""
lblArmyStat = ""
lblNavyStat = ""
lblTotalZips = ""
lblArmyRec = ""
lblNavyRec = ""

```

```

'open selected table
MIObj.Do "set next document parent " & pctDisplay.hWnd & "style 1"
'reopen user selected table
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
    "Us_zips.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
    "Us_hiway.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
    "States.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
    "Uszipbdy.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _

```

```

"MapArmy.tab" & Chr$(34) & " Interactive") 'Army stations in selected unit
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"MapNavy.tab" & Chr$(34) & " Interactive") 'Navy stations in selected unit
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"MapJoint.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"Ar_Rs.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"Nv_Rs.tab" & Chr$(34) & " Interactive")

If Service = "Army" Then
    MIObj.Do "Map From Uszipbdy, MapJoint, MapArmy, MapNavy, Ar_Rs, us_hiway, States"
Elseif Service = "Navy" Then
    MIObj.Do "Map From uszipbdy, MapJoint, MapArmy, MapNavy, Nv_Rs, us_hiway, States"
Elseif Service = "Joint" Then
    MIObj.Do "Map From Uszipbdy, MapJoint, MapArmy, MapNavy, us_hiway, States"
End If

'Get the handle to the front window
iMapInfoWinID = MIObj.eval("frontwindow()")

'set labels for zipcode
MIObj.Do ("Set Map Layer ""MapJoint"" Label Position Below Font
(""Arial"",257,9,16711680,16777215) With Zip_code Auto on")
MIObj.Do ("Set Map Layer ""MapArmy"" Label Position Below Font
(""Arial"",257,9,16711680,16777215) With Zip_code Auto on")
MIObj.Do ("Set Map Layer ""MapNavy"" Label Position Below Font
(""Arial"",257,9,16711680,16777215) With Zip_code Auto on")

cmdDoneGams_Exit:
Exit Sub

HandleDoneGamsError:

Select Case Err.Number
Case 53, 75 'file not found, path error
Resume Next
Case Else
MsgBox "Error Number: " + Str(Err.Number) + _
"; Description: " + Err.Description + _
".", vbInformation, _
"The Friendly Error Handler"
Resume Next
End Select

End Sub

Public Sub cmdDoneSetup_Click()
'Set screen view. Make Open,Close RS tab visible
OpenCloseTabs.Visible = False
cmdDoneGAMS.Visible = False
cmdDoneSetup.Visible = False
pctMeasure.Visible = True
cmdResetMetrics.Visible = True
cmdGoMain.Enabled = True
cmdRunGams.Enabled = True

```

```

cmdDisplayGAMS.Enabled = False
cmdSetupOptimizer.Enabled = True
cmdSetupOptimizer.Caption = "Setup Optimizer"
frmLegend.Visible = True
Call DisplayPanelMsg("Select 'Run Optimizer'")

'Reset ValidateZipCounter to 1
ValidateZipCount = 1

'Loop to clear control arrays for opening and closing
For iindex = 1 To 10
    txtAOpenCloseRS(iindex) = ""
    txtAopenRS(iindex) = ""
    txtAcloseRS(iindex) = ""
    txtNopenRS(iindex) = ""
    txtNcloseRS(iindex) = ""
    txtNOpenCloseRS(iindex) = ""
Next iindex

End Sub

Private Sub cmdExit_Click()
On Error GoTo HandleExitRslesError

If cmdDoneGAMS.Visible = True Then
    strMsg = "You must first select DONE."
    Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "User Message")
Exit Sub
Else

'Close MapInfo files before killing. Otherwise error #75 results
MIObj.Do "close all interactive"
Kill (cMapPath & "Map*.*)      'All MapInfo files

'close out objects and release all memory
Set MIObj = Nothing
Set datOptimize.Recordset = Nothing
Set datTempQuery.Recordset = Nothing
Set datTempQuery2.Recordset = Nothing
Set datTempQuery2.Recordset = Nothing
Set datValidateZip.Recordset = Nothing

End
End If

cmdFileExit_Exit:
Exit Sub

HandleExitRslesError:
Select Case Err.Number
    Case 75
        Resume Next
    Case Else
        MsgBox Err.Description, vbOKOnly + vbExclamation, "Unexpected Error"
        Resume Next
End Select

```

End Sub

Private Sub cmdSetupOptimizer_Click()

```
Dim OpenCloseRScout As Integer
Dim CloseRScout As Integer
Dim OpenRScout As Integer
Dim NOpenCloseRScout As Integer
Dim NCloseRScout As Integer
Dim NOpenRScout As Integer
Dim TotalCount As Integer
Dim OpenStatus As Integer
Dim CloseStatus As Integer
Dim CandidateStatus As Integer
Dim FileToSort As String
Dim FileToSort2 As String
Dim TotalProd, TotalProd2 As Integer 'Variable to hold unit production
Dim ZipMsg As String
```

On Error GoTo HandleOptimizeError

'delete files created for GAMS optimization

Kill (cGamsPath & "gamsin.*")

Kill (cGamsPath & "zzz.*")

Kill (cGamsPath & "armynavy2.lst")

Kill (cGamsPath & "station.txt")

'Set screen view. Make tab visible

cmdSetupOptimizer.Caption = "Continue Setup"

cmdDoneSetup.Visible = True

cmdDoneGAMS.Visible = False

cmdGoMain.Enabled = False

cmdDisplayGAMS.Enabled = False

cmdClear.Enabled = False

pctMeasure.Visible = False

cmdResetMetrics.Visible = False

'Set screen view

cmdDoneSetup.Visible = False

cmdClear.Enabled = True

OpenCloseTabs.Visible = True

frmLegend.Visible = False

If Service = "Army" Then

OpenCloseTabs.TabVisible(0) = True 'show Army tab

OpenCloseTabs.TabVisible(1) = False 'close Navy tab

OpenCloseTabs.TabVisible(2) = False 'close AF Tab

Elseif Service = "Navy" = True Then

OpenCloseTabs.TabVisible(0) = False

OpenCloseTabs.TabVisible(1) = True 'show Navy tab

OpenCloseTabs.TabVisible(2) = False

Elseif Service = "Joint" = True Then 'show Army & Navy tabs

OpenCloseTabs.TabVisible(0) = True

OpenCloseTabs.TabVisible(1) = True

OpenCloseTabs.TabVisible(2) = False

End If

```

If RunOptCount = 1 Then
'Loop to clear control arrays for opening and closing
For iindex = 1 To 10
    txtAOpenCloseRS(iindex) = ""
    txtAopenRS(iindex) = ""
    txtAcloseRS(iindex) = ""
    txtNOpenCloseRS(iindex) = ""
    txtNopenRS(iindex) = ""
    txtNcloseRS(iindex) = ""
Next iindex
Call DisplayPanelMsg("Enter zip codes in the tabs")
strMsg = " " & _
    "Please read before proceeding." & vbCrLf & _
    "    Do NOT enter the same zip code in two different text boxes." & vbCrLf & _
    "    " & _
    "    This will result in a invalid setup." & vbCrLf & _
    "Also, use the BACKSPACE key or 'Clear' button to remove unwanted entries," & _
    "    " & _
    "Press CONTINUE SETUP when finished."
Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Input Message")
RunOptCount = RunOptCount + 1
Else
End If

'Where should the map draw
MIObj.Do ("set next document parent " & pctDisplay.hWnd & " style 1")

'Initialize counters & status variables
ValidZipCode = True
RSCount = 0
OpenCloseRSCount = 0
OpenRSCount = 0
CloseRSCount = 0
NOpenCloseRSCount = 0
NOpenRSCount = 0
NCloseRSCount = 0
CloseStatus = 2 '0    'used in the gamsin.std file
OpenStatus = 2 '1    'used in the gamsin.std file
CandidateStatus = 2 'used in the gamsin.std file

'check all control arrays for valid user entries and size of each array
If Service = "Army" Or Service = "Joint" Then
    If RunOptCount > 1 Then
        ExitModule = False
    End If
    'Check control array for stations to open
    ZipMsg = "You did not enter any zip codes where Army stations should be opened." & vbCrLf
    & _
        "    Do you want to enter zip codes where stations should be opened?"
    Let RSstatus = "ArmyRSCount"
    OpenRSCount = FindArraySize(txtAopenRS, ZipMsg, RSstatus, UnitSelected)
    RSCount = 0 'reset to zero
    If ExitModule = True Then
        Exit Sub
    End If

```

```

'check control array for "candidate" stations
ZipMsg = "You did not enter any zip codes where Army stations should be closed." & vbCrLf &
-
    " Do you want to enter zip codes where stations should be closed?"
    Let RSstatus = "ArmyRSClose"
    CloseRSCount = FindArraySize(txtAcloseRS, ZipMsg, RSstatus, UnitSelected)
    RSCount = 0 'reset count to zero
    If ExitModule = True Then
        Exit Sub
    End If

'check control array for stations to open
ZipMsg = "No candidate zip codes were entered for Army stations. " & vbCrLf & _
    " Do you want to enter zip codes?"
    Let RSstatus = "ArmyRSOpenClose"
    OpenCloseRSCount = FindArraySize(txtAOpenCloseRS, ZipMsg, RSstatus, UnitSelected)
    RSCount = 0 'reset count to zero
    If ExitModule = True Then
        Exit Sub
    End If
End If

'Check Navy control arrays here
If Service = "Navy" Or Service = "Joint" Then
    If RunOptCount > 1 Then
        ExitModule = False
    End If
    'check control array for stations to open
    ZipMsg = "You did not enter any zip codes where Navy stations should be opened." & vbCrLf &
-
    " Do you want to enter zip codes where stations should be opened?"
    Let RSstatus = "NavyRSOpen"
    NOpenRSCount = FindArraySize(txtNopenRS, ZipMsg, RSstatus, UnitSelected)
    RSCount = 0 'reset to zero
    If ExitModule = True Then
        Exit Sub
    End If
    'check control array for stations to close
    ZipMsg = "You did not enter any zip codes where Navy stations should be closed." & vbCrLf &
-
    " Do you want to enter zip codes where stations should be closed?"
    Let RSstatus = "NavyRSClose"
    NCloseRSCount = FindArraySize(txtNcloseRS, ZipMsg, RSstatus, UnitSelected)
    RSCount = 0 'reset count to zero
    If ExitModule = True Then
        Exit Sub
    End If
    'check control array for "candidate" stations
    ZipMsg = "No candidate zip codes were entered for Navy stations. " & vbCrLf & _
    " Do you want to enter zip codes?"

    Let RSstatus = "NavyRSOpenClose"
    NOpenCloseRSCount = FindArraySize(txtNOpenCloseRS, ZipMsg, RSstatus, UnitSelected)
    RSCount = 0 'reset count to zero
    If ExitModule = True Then
        Exit Sub

```

End If
End If

Screen.MousePointer = vbHourglass

```
'=====Create Production
File=====
'determine production data for the selected year/quarter
'the fields are gsma_army and gsma_navy
Open cGamsPath & "gamsin.tgt" For Output As #1
If Service = "Army" Then
    If frmMain.optDistrict = True Then
        strSQL = "SELECT Sum(gsma_army) AS TotArmyProd, sum(gsma_navy) AS TotNavyProd "
        strSQL2 = "FROM Rslesdb WHERE Year = " & frmMain.cboYear & " and Quarter"
        strSQL3 = " = " & frmMain.cboQtr & " and ar_battn = " & UnitSelected & ""
    Else
        strSQL = "SELECT Sum(gsma_army) AS TotArmyProd, sum(gsma_navy) AS TotNavyProd "
        strSQL2 = "FROM Rslesdb WHERE Year = " & frmMain.cboYear & " and Quarter"
        strSQL3 = " = " & frmMain.cboQtr & " and ar_co = " & UnitSelected & ""
    End If
Elseif Service = "Navy" Then
    If frmMain.optDistrict = True Then
        strSQL = "SELECT Sum(gsma_army) AS TotArmyProd, sum(gsma_navy) AS TotNavyProd "
        strSQL2 = "FROM Rslesdb WHERE Year = " & frmMain.cboYear & " and Quarter"
        strSQL3 = " = " & frmMain.cboQtr & " and nv_nrd = " & UnitSelected & ""
    Else
        strSQL = "SELECT Sum(gsma_army) AS TotArmyProd, sum(gsma_navy) AS TotNavyProd "
        strSQL2 = "FROM Rslesdb WHERE Year = " & frmMain.cboYear & " and Quarter"
        strSQL3 = " = " & frmMain.cboQtr & " and nv_nrz = " & UnitSelected & ""
    End If
Elseif Service = "Joint" Then
    If frmMain.optMetro = True Then
        strSQL = "SELECT Sum(gsma_army) AS TotArmyProd, sum(gsma_navy) AS TotNavyProd "
        strSQL2 = "FROM Rslesdb WHERE Year = " & frmMain.cboYear & " and Quarter"
        strSQL3 = " = " & frmMain.cboQtr & " and metro = " & UnitSelected & ""
    End If
End If
```

datOptimize.RecordSource = strSQL + strSQL2 + strSQL3
datOptimize.Refresh

```
'Select correct total production message
If Service = "Army" Then
    strMsg = "    Annual Production data for " & UnitSelected & " was " & _
        Int(datOptimize.Recordset!totArmyProd) & "." & vbCrLf & _
        "Do you want to use this figure for the MinCost Optimizer?"
Elseif Service = "Navy" Then
    strMsg = "    Annual Production data for " & UnitSelected & " was " & _
        Int(datOptimize.Recordset!totNavyProd) & "." & vbCrLf & _
        "Do you want to use this figure for the MinCost Optimizer?"
Elseif Service = "Joint" Then
    strMsg = "    Annual Production data:  Army -- " & _
        (Int(datOptimize.Recordset!totArmyProd) & _
        " Navy -- " & Int(datOptimize.Recordset!totNavyProd)) & _
        & "." & vbCrLf & _
```

```

" & vbCrLf & _
"Do you want to use these figures for the MinCost Optimizer?"
End If

iResponse = MsgBox(strMsg, vbYesNo + vbQuestion, "Input Message")
If iResponse = vbNo Then 'Let user input target production
    If Service = "Army" Then
        Let TotalProd = InputBox("Enter target production: ")
        Print #1, "army" & Space(3) & TotalProd
        Print #1, "navy" & Space(3) & Int(datOptimize.Recordset!totNavyProd)
    ElseIf Service = "Navy" Then 'enter Navy user prod target
        Let TotalProd = InputBox("Enter target production: ")
        Print #1, "army" & Space(3) & Int(datOptimize.Recordset!totArmyProd)
        Print #1, "navy" & Space(3) & TotalProd
    ElseIf Service = "Joint" Then 'enter Army & Navy user prod target
        Let TotalProd = InputBox("Enter Army target production: ")
        Let TotalProd2 = InputBox("Enter Navy target production: ")
        Print #1, "army" & Space(3) & TotalProd
        Print #1, "navy" & Space(3) & TotalProd2
    End If
Else
    Print #1, "army" & Space(3) & Int(datOptimize.Recordset!totArmyProd)
    Print #1, "navy" & Space(3) & Int(datOptimize.Recordset!totNavyProd)
End If
Close #1 'close gamsin.tgt file

Call DisplayPanelMsg(BuildGamsFilesMSG)

'If program gets here all data is valid=====
'Write the Zip codes to be closed, opened, or candidates to a file
'Open the 4 required files that will be input into GAMS (Army & Navy stations &
' where there is a station)
Open cGamsPath & "gamsin.sti" For Output As #2 'zips with stations
Open cGamsPath & "zzz.sti" For Output As #3 'zips with stations listed twice
Open cGamsPath & "gamsin.std" For Output As #4 'zips + attributes
Open cGamsPath & "gamsin.cst" For Output As #5 'zips + cost data

'Print header line for gamsin.std
Print #4, Tab(9); "llong"; Tab(18); "llat"; Tab(27); "arec"; Tab(32); "nrec"; _
    Tab(39); "lpop"; Tab(47); "astatus"; Tab(55); "nstatus"
Print #5, Tab(9); "coststa"; Tab(18); "costusa"; Tab(27); "costusn"; Tab(37); "costrec"; _
    Tab(47); "costj2"

strSQL = "SELECT DISTINCT zip_code, astatzip, nstatzip, Llong,Llat,Opra_Ar,
Oprn_Nv,CostSta,Costusa,Costusn,CostRec,Costj2,Pop17 FROM Rslesdb WHERE (astatzip =1
or nstatzip = 1) "
strSQL2 = "and Year = " & Trim(frmMain.cboYear) & " and Quarter"
strSQL3 = " = " & frmMain.cboQtr & " and " & unit & " = " & UnitSelected & "" 'add quarter
With datOptimize
    .RecordSource = strSQL & strSQL2 + strSQL3
    .Refresh
    TotalCount = .Recordset.RecordCount
    For iindex = 1 To TotalCount
        Print #2, .Recordset!Zip_Code
        Print #3, .Recordset!Zip_Code; Tab(8); datOptimize.Recordset!Zip_Code
        Print #4, .Recordset!Zip_Code; Tab(9); _

```

```

Format(datOptimize.Recordset!Llong, "###.000"); _
Tab(18); Format(.Recordset!Llat, "##.000"); _
Tab(26); .Recordset!opra_ar; _
Tab(31); .Recordset!Oprn_nv; _
Tab(38); .Recordset!pop17;
If .Recordset!astatzip = 1 Then 'print correct army status (a 1 indicates a station is
currently there)
    Print #4, Tab(46); OpenStatus; 'open status is a 1
Else
    Print #4, Tab(46); CloseStatus; 'CloseStatus;
End If
If .Recordset!nstatzip = 1 Then 'print correct navy status
    Print #4, Tab(54); OpenStatus
Else
    Print #4, Tab(54); CloseStatus 'CloseStatus
End If

'Print .cst file info (for all currently open stations only)
Print #5, .Recordset!Zip_Code; _
Tab(8); .Recordset!coststa; _
Tab(17); .Recordset!costusa; _
Tab(26); .Recordset!costusn; _
Tab(36); .Recordset!costrec; _
Tab(46); .Recordset!costj2

.Recordset.MoveNext
Next iindex
End With
Close #2, #3, #4, #5
Set datOptimize.Recordset = Nothing

'Write info to .std file from control arrays (sort it later)
If Service = "Army" Or frmMain.optJoint = True Then
    Call WriteSTDfile(txtAopenRS, OpenRScount, 1, "A") 'ArmyRSopen
    Call WriteCSTfile(txtAopenRS, OpenRScount)
    Call WriteSTDfile(txtAcloseRS, CloseRScount, 0, "A") 'ArmyRSclose
    Call WriteCSTfile(txtAcloseRS, CloseRScount)
    Call WriteSTDfile(txtAOpenCloseRS, OpenCloseRScount, 2, "A") 'ArmyRSOpenClose-free
    Call WriteCSTfile(txtAOpenCloseRS, OpenCloseRScount)
End If
If Service = "Navy" Or Service = "Joint" Then
    Call WriteSTDfile(txtNopenRS, NOpenRScount, 1, "N") 'NavyRSopen
    Call WriteCSTfile(txtNopenRS, NOpenRScount)
    Call WriteSTDfile(txtNcloseRS, NCloseRScount, 0, "N") 'NavyRSclose
    Call WriteCSTfile(txtNcloseRS, NCloseRScount)
    Call WriteSTDfile(txtNOpenCloseRS, NOpenCloseRScount, 2, "N") 'NavyRSOpenClose-free
    Call WriteCSTfile(txtNOpenCloseRS, NOpenCloseRScount)
End If

Call SortSTDfile(cGamsPath & "gamsin.std") 'sort the completed .std file
Call SortCSTfile(cGamsPath & "gamsin.cst") 'sort the completed .cst file
'send files to array, sort, then check for duplicates. Then rewrite files
FileToSort = cGamsPath & "gamsin.sti" 'Zips w/stations
FileToSort2 = cGamsPath & "zzz.sti" 'Zips w/stations twice
Call SortZZZfiles(FileToSort, FileToSort2) 'function to sort zip code files

```

```

'Create domain files for GAMS (gamsin.zpd, gamsin.zpi, zzz.zpi)
Open cGamsPath & "gamsin.zpi" For Output As #4 'Zips in the domain
Open cGamsPath & "zzz.zpi" For Output As #5 'Zips in the Domain (listed twice)
Open cGamsPath & "gamsin.zpd" For Output As #6 'Attributes of domain
'Print header line for gamsin.zpd file
Print #6, Tab(9); "long"; Tab(19); "lat"; Tab(27); "pop"; Tab(33); "hs1"; _
    Tab(39); "hs2"; Tab(45); "azip"; Tab(51); "nzip"; Tab(57); "jzip"; _
    Tab(63); "area"; Tab(71); "density"; Tab(81); "income"; Tab(88); "urate"; _
    Tab(97); "urban"; Tab(104); "suburb"
'Determine correct query to write .zpd file
strSQL = "SELECT DISTINCT zip_code,Llong,Llat, Pop17, hs1, hs2, astatzip, nstatzip, jstatzip,"
strSQL2 = "area, density, income, urate, urban, suburb FROM Rslesdb "
'strSQL3 = "WHERE " & unit & " = " & UnitSelected & ""
strSQL3 = "WHERE Year = " & frmMain.cboYear & " and Quarter = " & frmMain.cboQtr & " and "
& unit & " = " & UnitSelected & ""
'Write .zpi and .zpd files
strSQL = strSQL + strSQL2 + strSQL3

TotalCount = 0
With datOptimize
    .RecordSource = strSQL
    .Refresh
    Let TotalCount = .Recordset.RecordCount
    For iindex = 1 To TotalCount
        .Recordset.RecordCount
        Print #4, .Recordset!Zip_Code
        Print #5, .Recordset!Zip_Code; Tab(8); .Recordset!Zip_Code
        Print #6, .Recordset!Zip_Code; Tab(9); _
            Format(Val(.Recordset!Llong), "###.000"); _
            Tab(19); Format(Val(.Recordset!Llat), "##.000"); _
            Tab(26); Val(.Recordset!pop17); _
            Tab(32); Val(.Recordset!hs1); _
            Tab(38); Val(.Recordset!hs2); _
            Tab(44); Val(.Recordset!astatzip); _
            Tab(50); Val(.Recordset!nstatzip); _
            Tab(56); Val(.Recordset!JStatZip); _
            Tab(63); Format(Val(.Recordset!area), "###.00"); _
            Tab(71); Format(Val(.Recordset!density), "####.00"); _
            Tab(80); Val(.Recordset!income); _
            Tab(88); Format(Val(.Recordset!urate), "##.00"); _
            Tab(96); Val(.Recordset!urban); _
            Tab(103); Val(.Recordset!suburb)
        .Recordset.MoveNext
    Next iindex
End With
Close #4, #5, #6

Screen.MousePointer = vbDefault

'reset RunOptCount to initial value
RunOptCount = 1

'Notify user to select Run Optimizer
strMsg = "Optimizer setup files are complete. Select 'Run Optimizer' to optimize."
Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "User Message")

'close the Optimizer screen view

```

```

cmdDoneSetup_Click

cmdSetupOptimizer_exit:
Exit Sub

HandleOptimizeError:
Select Case Err.Number
Case 53, 55, 75 'file not found, file already open, path not found
Resume Next
Case 9 'subscript out of range
Resume Next
Case 94 'invalid use of null
Call DisplayPanelMsg(RecordsetFailMSG)
Resume Next
Case Else
MsgBox "Error Number: " + Str(Err.Number) + _
"; Description: " + Err.Description + _
".", vbInformation, _
"The Friendly Error Handler"
strMsg = "This may cause the optimizer to fail."
Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Error")
Resume Next
End Select
Resume
End Sub
Public Function SortZZZfiles(ByVal FileToSort As String, FileToSort2 As String)
*****
'* This function sorts the 2 zip code files and eliminates duplicate zip codes
'* Read gamsin.sti into an array, sort, eliminate duplicates, and count. Loop
'* through counter, printing 1 zip to gamsin.sti and 2 zips to zzz.sti
'*
*****
Dim OptArray() As Single
Dim SortCount As Integer
Dim iRow, icol, x As Integer
Dim passNum As Integer
Dim temp As Single

Open FileToSort For Input As #1
SortCount = 0
Do While Not (EOF(1))
Input #1, InData
' increment record counter by 1
SortCount = SortCount + 1
Loop
Close #1
ReDim OptArray(1 To SortCount) As Single

'Fill the array with the zip codes
Open FileToSort For Input As #1
For iRow = 1 To SortCount
Input #1, OptArray(iRow)
Next iRow
Close #1

'sort zips in ascending order

```

```

Open FileToSort For Input As #1
For passNum = 1 To (SortCount - 1)
  For x = 1 To (SortCount - passNum)
    If OptArray(x) > OptArray(x + 1) Then 'switch them
      Let temp = OptArray(x)
      Let OptArray(x) = OptArray(x + 1)
      Let OptArray(x + 1) = temp
    End If
  Next x
Next passNum
Close #1

'remove duplicate zip codes
Open FileToSort For Output As #1
Open FileToSort2 For Output As #2
For x = 1 To SortCount
  If x = SortCount Then
    GoTo PrintLastZip
  End If

  If OptArray(x) = OptArray(x + 1) Then
    If Len(CStr(OptArray(x + 1))) = 4 Then
      Print #1, "0" & CStr(OptArray(x + 1))
      Print #2, "0" & CStr(OptArray(x + 1)); Tab(9); "0" & CStr(OptArray(x + 1))
    Else
      Print #1, CStr(OptArray(x + 1)) 'Print zip
      Print #2, CStr(OptArray(x + 1)); Tab(9); CStr(OptArray(x + 1))
    End If
    x = x + 1
  Else
    PrintLastZip:
    If Len(CStr(OptArray(x))) = 4 Then
      Print #1, "0" & CStr(OptArray(x))
      Print #2, "0" & CStr(OptArray(x)); Tab(9); "0" & CStr(OptArray(x))
    Else
      Print #1, CStr(OptArray(x)) 'Print zip
      Print #2, CStr(OptArray(x)); Tab(9); CStr(OptArray(x))
    End If
    'End if 'shouldn't there be one here???
    ' If x = SortCount Then
    '   Exit For
    ' Else
    '   Do While (x <= SortCount - 1) And (OptArray(x) = OptArray(x + 1))
    '     x = x + 1
    '   Loop
    ' End If
  End If

  If x = SortCount Then
    Exit For
  Else
    ' Do While (i <= iCount - 1) And (iZip(i) = iZip(i + 1))
    '   i = i + 1
    ' Loop
  End If
Next x

```

```

Close #1, #2
End Function

```

```

Private Function IsActive(hprog) As Long
    Dim hProc, RetVal As Long
    Const PROCESS_QUERY_INFORMATION = &H400
    Const STILL_ACTIVE = 259

    hProc = OpenProcess(PROCESS_QUERY_INFORMATION, False, hprog)
    If hProc <> 0 Then
        GetExitCodeProcess hProc, RetVal
    End If

    IsActive = (RetVal = STILL_ACTIVE)
    CloseHandle hProc

```

```

End Function

```

```

Private Sub cmdDisplayGAMS_Click()

```

```

    Dim iRow, iCol As Integer    'indexes to read in data from GAMS output file
    Dim totArmySta, totArmyRec, totNavySta, totNavyRec, totJointSta As Single
    Dim A_StaZip, N_StaZip, totArmyProd, totNavyProd As Single
    Dim GetNewFileName, gstGetNewFileName As String
    Dim headerLine As String
    Dim GamsInArray() As Single 'dynamic, multidimensional array

```

```

    Screen.MousePointer = vbHourglass
    On Error GoTo HandleErrors

```

```

'Set screen parameters (enable DONE button, disable others)
cmdDoneGAMS.Visible = True
pctGAMSoutput.Visible = True
frmLegend.Visible = True
OpenCloseTabs.Visible = False
pctMeasure.Visible = False
cmdResetMetrics.Visible = False
cmdDisplayGAMS.Enabled = True 'change this to false when done
cmdSetupOptimizer.Enabled = False
cmdClear.Enabled = False
cmdGoMain.Enabled = False

```

```

'Notify user to select Run Optimizer
strMsg = "This process may take several minutes to complete."
Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "User Message")

```

```

Call DisplayPanelMsg("Displaying Optimizer Output")

```

```

'Fill Pre-GAMS Summary Data Box
strSQL = "SELECT Sum(AStatZip) - Sum(JStatZip) AS TotalArmyStations, Sum(NStatZip) - Sum(JStatZip) AS TotalNavyStations, Sum(JStatZip) AS TotalJointStations, Sum([Opra_ar]) AS TotalArmyRec, Sum([Oprn_nv]) AS TotalNavyRec, Sum([Gsma_army]) AS TotalArmyProd, Sum([Gsma_navy]) AS TotalNavyProd, Count([Zip_Code]) As TotalZips From RSLESDB WHERE " & unit & " like " & UnitSelected & " and year = " & frmMain.cboYear.Text & " and quarter = " & frmMain.cboQtr.Text & " GROUP BY " & unit & ""

```

```

datTempQuery.RecordSource = strSQL
datTempQuery.Refresh
With datTempQuery
    'Display totals in Pre-GAMS Summary display box
    Let lblAsIsUnit = Service & " Unit " & UnitSelected
    Let lblPreJointSta = .Recordset!TotalJointStations
    Let lblPreArmySta = .Recordset!TotalArmyStations
    Let lblPreNavySta = .Recordset!TotalNavyStations
    Let lblPreArmyRec = .Recordset!TotalArmyRec
    Let lblPreNavyRec = .Recordset!TotalNavyRec
    Let lblPreZipTotal = .Recordset!TotalZips
    Let lblPreArmyProd = .Recordset!TotalArmyProd
    Let lblPreNavyProd = .Recordset!TotalNavyProd
End With

GetNewFileName = "" 'set return filename to empty string

GetNewFileName = (cGamsPath & "stations.txt")
Open GetNewFileName For Input As #1
    'determine the number of lines in the file
    Line Input #1, headerLine ' skip header line
    iCount = 0
    Do While Not (EOF(1))
        For icol = 1 To 12
            Input #1, InData 'read the 12 fields from gams output file
            Next icol
            ' increment record counter by 1
            iCount = iCount + 1
        Loop
    Close #1
    ReDim GamsInArray(1 To iCount, 1 To 12) As Single

Open GetNewFileName For Input As #1
    'populate the array
    Line Input #1, headerLine ' skip header line
    For iRow = 1 To iCount
        For icol = 1 To 12
            Input #1, GamsInArray(iRow, icol)
        Next icol
    Next iRow

Close #1

'Determine total Army or Navy, and Joint Stations and # recruiters
'initialize Station & Recruiter counters
totArmySta = 0
totArmyRec = 0
totNavySta = 0
totNavyRec = 0
totJointSta = 0
totArmyProd = 0
totNavyProd = 0

For iRow = 1 To iCount
    totArmyRec = totArmyRec + GamsInArray(iRow, 5)
    totNavyRec = totNavyRec + GamsInArray(iRow, 6)

```

```

totArmyProd = totArmyProd + GamsInArray(iRow, 9)
totNavyProd = totNavyProd + GamsInArray(iRow, 10)
If GamsInArray(iRow, 4) = 1 Then
    totJointSta = totJointSta + GamsInArray(iRow, 4)
Else
    totArmySta = totArmySta + GamsInArray(iRow, 2)
    totNavySta = totNavySta + GamsInArray(iRow, 3)
End If
Next iRow

'Display totals in GAMS Summary display box
Let lblJointStat = totJointSta
Let lblArmyStat = totArmySta
Let lblNavyStat = totNavySta
Let lblArmyRec = totArmyRec
Let lblNavyRec = totNavyRec
Let lblTotalZips = iCount
Let lblUnitSummary = Service & " Unit " & UnitSelected
Let lblArmyProd = Format(totArmyProd, "##,###.0")
Let lblNavyProd = Format(totNavyProd, "##,###.0")

On Error Resume Next

'Read in the GAMS text file to STATION.TAB
MIObj.Do "Register Table " & Chr$(34) & GetNewFileName & Chr$(34) & "Type ASCII Delimiter
44 " _
    & "Titles Charset ""Windowslatin1"" Into ""c:\Program
Files\Rsles\MapInfoTables\Station.TAB""
MIObj.Do "Open Table ""c:\Program Files\Rsles\MapInfoTables\Station.TAB"" Interactive"

'Make copies of table station into AStation & NStation and open tables
MIObj.Do "Commit Table Station As ""c:\Program Files\Rsles\MapInfoTables\AStation.tab"" TYPE
NATIVE Charset " _
    & ""WindowsLatin1""
MIObj.Do "Open Table ""c:\Program Files\Rsles\MapInfoTables\AStation.TAB"" Interactive"
MIObj.Do "Commit Table Station As ""c:\Program Files\Rsles\MapInfoTables\NStation.tab""
TYPE NATIVE Charset " _
    & ""WindowsLatin1""
MIObj.Do "Open Table ""c:\Program Files\Rsles\MapInfoTables\NStation.TAB"" Interactive"

'geocode the 3 files just created
Call Geocode("NStation", "R") 'red ship
Call Geocode("AStation", "B") 'brown tank
Call Geocode("Station", "Y") 'yellow star-joint stations

'Queries to create JOINT, ARMY, and NAVY tables
MIObj.Do "Select station.zip_code, station.armyrec, station.navyrec, " & _
    "station.armyrsprod, station.navyrsprod FROM Station WHERE JointSta=1 into Joint"
MIObj.Do "Select nstation.zip_code, nstation.navyrec, nstation.navyrsprod " & _
    "FROM NStation WHERE NavySta=1 and ArmySta = 0 into Navy"
MIObj.Do "Select astation.zip_code, astation.armyrec, astation.armyrsprod " & _
    "FROM AStation WHERE ArmySta=1 and NavySta = 0 into Army"

Screen.MousePointer = vbDefault

MIObj.Do ("set next document parent " & pctDisplay.hWnd & " style 1") 'added this because

```

```
'it wouldn't display the results twice in a row
iMapInfoWinID = MIObj.eval("frontwindow()")
```

```
MIObj.Do ("Add Map Layer Army, Navy, Joint")
'army layer label properties
MIObj.Do ("Set map layer 1 label with ""Zip:"" + zip_code + Chr$(13) + ""Rec:"" + " _
& " armyRec + Chr$(13) + ""Prod:"" + ArmyRSProd Auto on")
'navy layer label properties
MIObj.Do ("Set map layer 2 label with ""Zip:"" + zip_code + Chr$(13) + ""Rec:"" + " _
& " navyRec + Chr$(13) + ""Prod:"" + NavyRSProd Auto on")
'joint layer label properties
MIObj.Do ("Set map layer 3 label with ""Zip:"" + zip_code + Chr$(13) + ""ARec:"" + " _
& " armyRec + Chr$(13) + ""AProd:"" + armyrsprod + chr$(13) + ""NRec:"" + " _
& " navyRec + Chr$(13) + ""NProd:"" + navyrsprod Auto on")
```

```
Call sndPlaySound("c:\Program Files\Rsles\wow.wav", &H1)
```

```
Call DisplayPanelMsg("Select 'DONE' to return to original view.")
```

```
cmdDisplayGAMS_Exit:
Exit Sub
```

```
HandleErrors:
```

```
Select Case Err.Number
```

```
Case 53, 75 'file not found, path error
```

```
MsgBox "Error Number:" + Str(Err.Number) + _
```

```
" ; Description: " + Err.Description + _
```

```
".", vbInformation, _
```

```
"The Friendly Error Handler"
```

```
strMsg = "Ensure that the optimizer returned a feasible solution before displaying results."
```

```
Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Error Message")
```

```
Call cmdDoneGAMS_Click
```

```
Exit Sub
```

```
Case 3004, 3024, 3044
```

```
gstGetNewFileName = GetNewFileName
```

```
If gstGetNewFileName = "" Then 'User cancelled operation
```

```
Exit Sub
```

```
End If
```

```
Case Else
```

```
MsgBox "Error Number:" + Str(Err.Number) + _
```

```
" ; Description: " + Err.Description + _
```

```
".", vbInformation, _
```

```
"The Friendly Error Handler"
```

```
Resume Next
```

```
End Select
```

```
End Sub
```

```
Private Sub cmdResetMetrics_Click()
```

```
On Error GoTo HandleResetMetrics_Error
```

```
'needs error trapping for error #53 (file not found)
```

```
MIObj.Do ("close Table MapMetrics interactive")
```

```
MIObj.Do ("close Table MapMetrics1 interactive")
```

```
Kill (cMapPath & "MapMetrics.*")
```

```
Kill (cMapPath & "MapMetrics1.*")
```

```

'remove thematic layer
If Service = "Army" Or Service = "Navy" Then
    'The thematic layer always shows up as layer 5 for individual services
    MIObj.Do ("Remove Map Layer 5")
End If

DisplayPanelMsg ("Select Metric, View Data, or Run Optimizer")

cmdResetMetrics_Exit:
Exit Sub

HandleResetMetrics_Error:

    Select Case Err.Number
        Case 53, 75 'file not found
            Resume Next
        Case Else
            MsgBox "Error Number: " + Str(Err.Number) + _
                "; Description: " + Err.Description + _
                ". ", vbInformation, _
                "The Friendly Error Handler"
            Resume Next
        End Select
    End Sub

Private Sub cmdRunGams_Click()

'Display panel msg
Call DisplayPanelMsg("Running GAMS...was there a feasible solution?")

'Display Warning Message
Call DisplayMsg("
    " & _
    "The Optimizer will now optimize " & UnitSelected & ". " & vbCrLf & _
    "    This may take awhile to complete! " & vbCrLf & _
    " Be sure to observe if the optimizer returns a feasible solution." & vbCrLf & _
    "You will be unable to use this program until the process is complete.", vbCrLf & _
    vbExclamation, vbOKOnly, "WARNING")

Screen.MousePointer = vbHourglass

'Shell to GAMS optimizer
hprog = Shell("c:\Program Files\Rsles\gams\runit.bat", vbMaximizedFocus) 'returns taskID
Do While IsActive(hprog)
    DoEvents
Loop
'Get Process handle
hProc = OpenProcess(PROCESS_ALL_ACCESS, False, hprog)
'wait until the process terminates
If hProc <> 0 Then
    RetVal = WaitForSingleObject(hProc, INFINITE)
    CloseHandle hProc
End If

Screen.MousePointer = vbDefault
strMsg = "I'm back from Optimizing!"

```

```

Call DisplayMsg(strMsg, vbExclamation, vbOKOnly, "RSLES")

strMsg = "Did the GAMS run result in a feasible solution?"
iResponse = MsgBox(strMsg, vbYesNo + vbQuestion, "Input Message")
If iResponse = vbYes Then    'Provide user directions to run optimizer
    Call DisplayMsg("Select 'VIEW OPTIMIZER OUTPUT' to view the results.", _
        vbExclamation, vbOKOnly, "OUTPUT MESSAGE")
Else
    Call DisplayMsg("    There was an error during GAMS processing." & vbCrLf & _
        "    See 'GAMS errors' in the HELP menu." & vbCrLf & vbCrLf & _
        "    Possible changes to input include: " & vbCrLf & _
        "    1. 'Freeing Up' more stations for the optimizer." & vbCrLf & _
        "    2. Adding additional stations to consider for opening." & vbCrLf & _
        "    3. Lowering the production goals. " & vbCrLf & _
        "    4. You may have entered the same zip code twice. " & vbCrLf & vbCrLf & _
        "After making one or more of the changes above, re-run GAMS.", _
        vbExclamation, vbOKOnly, _
        "ERROR MESSAGE")
End If

cmdSetupOptimizer.Enabled = True
cmdSetupOptimizer.Caption = "Setup Optimizer"
cmdDisplayGAMS.Enabled = True

End Sub

Private Sub cmdToggle_Click()
    If pctRSArmy.Visible = True Then
        pctRSNavy.Visible = True
        pctNavyDistrict.Visible = True
        pctNavyZone.Visible = True
        pctRSArmy.Visible = False
        pctArmyBN.Visible = False
        pctArmyCO.Visible = False
    Else    'Navy boxes are currently visible
        pctRSArmy.Visible = True
        pctArmyBN.Visible = True
        pctArmyCO.Visible = True
        pctRSNavy.Visible = False
        pctNavyZone.Visible = False
        pctNavyDistrict.Visible = False
    End If
End Sub

End Sub

Private Sub cmdViewAsIsData_Click()
    pctGAMSoutput.Visible = False 'estimated data from GAMS
    pctAsIsData.Visible = True    'as is data box
End Sub

Private Sub cmdViewEstData_Click()
    pctGAMSoutput.Visible = True  'estimated data
    pctAsIsData.Visible = False  'as is data
End Sub

Private Sub Form_Activate()

```

```

Dim cMapFileName As String

' test to see if we should be re-drawing the map
'If g_bRefreshMap = False Then
'    Exit Sub
'End If

'set it so that we don't reMake tables (this will get set by frmMain.Continue button)
'g_bRefreshMap = False

'Initialize Counters for Run GAMS & ValidateZip Function
RunOptCount = 1      'initialize counter for Optimizer module
ValidateZipCount = 1 'set to 1 so it only performs SQL query once

On Error GoTo HandleFormActivateError
If DoneAbout = False Then 'if not returning from viewing "Help > About", load

'Set the screen view
pctMeasure.Visible = True
cmdRunGams.Enabled = False
cmdDisplayGAMS.Enabled = False
frmLegend.Visible = True
'display RS/CO/Zone/BN/District data boxes
cmdResetMetrics.Visible = True
If Service = "Army" Or Service = "Joint" Then
    pctRSArmy.Visible = True
    pctArmyBN.Visible = True
    pctArmyCO.Visible = True
ElseIf Service = "Navy" Then
    pctNavyDistrict.Visible = True
    pctRSNavy.Visible = True
    pctNavyZone.Visible = True
End If

'close all windows (make sure that a window exists
If MIObj.eval("frontwindow()") <> 0 Then
    MIObj.Do ("close window frontwindow()")
    MIObj.Do ("Close All")
End If

Screen.MousePointer = vbHourglass
'Tell MapInfo who owns it
MIObj.Do ("Set application window " & hWnd)

'dimension several variables in MapInfo space
MIObj.Do ("Dim aRowid as alias, aAddress As Alias, iCounter as integer, iResult As Integer, x as
float,y As Float")
MIObj.Do ("Dim symPoint As Symbol")

'Create recordset for unit selected by user in Main Form
'NOTE: will need to add production, areaname to SQL when available

'select info for army stations in selected unit
strSQL = "SELECT DISTINCT zip_code, " & RSname & " FROM Rslesdb "
strSQL2 = "WHERE astatzip = 1 and " & unit & " like " & UnitSelected & " and year = " &
frmMain.cboYear.Text & " and quarter = " & frmMain.cboQtr.Text

```

```

'select info for navy stations in selected unit
strSQL3 = "SELECT DISTINCT zip_code, " & RSname2 & " FROM Rslesdb "
strSQL4 = "WHERE nstatzip = 1 and " & unit & " like " & UnitSelected & " and year = " &
frmMain.cboYear.Text & " and quarter = " & frmMain.cboQtr.Text
'select info for joint stations in selected unit
strSQL5 = "SELECT DISTINCT zip_code, ar_rs, nv_rs FROM Rslesdb "
strSQL6 = "WHERE JstatZip = 1 and " & unit & " like " & UnitSelected & " and year = " &
frmMain.cboYear.Text & " and quarter = " & frmMain.cboQtr.Text

'Code to create dynaset, copy to a file, pass into MapInfo
datTempQuery.RecordSource = strSQL + strSQL2 'all zips for army stations
datTempQuery2.RecordSource = strSQL3 + strSQL4 'all zips for navy stations
datTempQuery3.RecordSource = strSQL5 + strSQL6 'all zips for joint stations

datTempQuery.Refresh
datTempQuery.Recordset.Sort = "zip_code Asc" 'Ascending sort
datTempQuery2.Refresh
datTempQuery2.Recordset.Sort = "zip_code Asc"
datTempQuery3.Refresh
datTempQuery3.Recordset.Sort = "zip_code Asc"

cMapFileName = "MapArmy" 'All army stations in selected unit
Call QueryConvert(datTempQuery.Recordset, cMapFileName)
cMapFileName = "MapNavy" 'all navy stations in selected unit
Call QueryConvert(datTempQuery2.Recordset, cMapFileName)
cMapFileName = "MapJoint" 'Navy stations in Metro
Call QueryConvert(datTempQuery3.Recordset, cMapFileName)

'Open the tables
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"Us_zips.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"Us_hiway.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"Uszipbdy.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"States.tab" & Chr$(34) & " Interactive")
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"MapArmy.tab" & Chr$(34) & " Interactive") 'Army stations in selected unit
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"MapNavy.tab" & Chr$(34) & " Interactive") 'Navy stations in selected unit
MIObj.Do ("Open table " & Chr$(34) & cMapPath & _
"MapJoint.tab" & Chr$(34) & " Interactive") 'MapJoint are army/navy stations

If Service = "Army" Then
MIObj.Do "Open Table ""c:\Program Files\Rsles\MapInfoTables\Ar_Rs.TAB"" Interactive"
ElseIf Service = "Navy" Then
MIObj.Do "Open Table ""c:\Program Files\Rsles\MapInfoTables\Nv_Rs.TAB"" Interactive"
ElseIf Service = "Joint" Then
MIObj.Do "Open Table ""c:\Program Files\Rsles\MapInfoTables\Metrobdy.TAB"" Interactive"
End If

'geocode army,navy,Joint tables
Call Geocode("MapArmy", "A") 'geocode MapArmy table w/Us_zips
Call Geocode("MapNavy", "N") 'geocode MapNavy table w/Us_zips
Call Geocode("MapJoint", "J") 'geocode table MapJoint w/Us_zips

```

```

'Where should the map draw
MIObj.Do ("set next document parent " & pctDisplay.hWnd & " style 1")
' if the joint table is mappable
If MIObj.eval("tableinfo(MapArmy,5)") = "T" Or MIObj.eval("tableinfo(MapNavy,5)") = "T" Then
  If MIObj.eval("tableinfo(MapJoint,5)") = "T" Then
    If Service = "Army" Then
      MIObj.Do "Map From Uszipbdy, MapJoint, MapArmy, MapNavy, Ar_Rs, us_hiway, States"
    ElseIf Service = "Navy" Then 'Navy
      MIObj.Do "Map From Uszipbdy, MapJoint, MapArmy, MapNavy, Nv_Rs, us_hiway, States"
    ElseIf Service = "Joint" Then
      MIObj.Do "Map From Uszipbdy, MapJoint, MapArmy, MapNavy, Metrobdy, States"
    End If

  Else
    If Service = "Army" Then
      MIObj.Do "Map From Uszipbdy, MapArmy, MapNavy, Ar_Rs, us_hiway, States"
    ElseIf Service = "Navy" Then
      MIObj.Do "Map From Uszipbdy, MapArmy, MapNavy, Nv_Rs, us_hiway, States"
    ElseIf Service = "Joint" Then
      MIObj.Do "Map From Uszipbdy, MapArmy, MapNavy, Metrobdy, us_hiway, States"
    End If
  End If
Else
  If MIObj.eval("tableinfo(MapJoint,5)") = "T" Then
    If Service = "Army" = True Then
      MIObj.Do "Map From Uszipbdy, MapJoint, Ar_Rs, us_hiway, States"
    ElseIf Service = "Navy" Then 'Navy
      MIObj.Do "Map From Uszipbdy, MapJoint, MapNavy, Nv_Rs, us_hiway, States"
    ElseIf Service = "Joint" Then 'Joint
      MIObj.Do "Map From Uszipbdy, MapJoint, MapArmy, MapNavy, Metrobdy, us_hiway, States"
    End If
  End If
End If

'Get the handle to the front window
iMapInfoWinID = MIObj.eval("frontwindow()")

Screen.MousePointer = vbDefault

'set labels for zipcode
MIObj.Do ("Set Map Layer ""MapJoint"" Label Position Below Font
(""Arial"",257,9,16711680,16777215) With Zip_code Auto on")
MIObj.Do ("Set Map Layer ""MapArmy"" Label Position Below Font
(""Arial"",257,9,16711680,16777215) With Zip_code Auto on")
MIObj.Do ("Set Map Layer ""MapNavy"" Label Position Below Font
(""Arial"",257,9,16711680,16777215) With Zip_code Auto on")

Else
  DoneAbout = False 'reset boolean variable to false so form is required to load
End If

Form_Activate_Exit:
Exit Sub

HandleFormActivateError:

```

```

Select Case Err.Number
    Case 70 'permission denied
        Resume Next
    Case 94 'invalid use of null
        Call DisplayPanelMsg(RecordsetFailMSG)
        Resume Next
    Case 3004, 3024, 3044 'database, file, or path not found
        MsgBox Err.Description
        mnuFileExit_Click
        Resume Next
    Case 524
        Resume Next
    Case Else
        Resume Next
End Select

End Sub

Private Sub SetRecordNumber()
'Display the record number
Dim iRecordCount As Double
Dim iCurrentRecord As Double

If Service = "Army" = True Then
    iRecordCount = datArmyRS.Recordset.RecordCount
    iCurrentRecord = datArmyRS.Recordset.AbsolutePosition + 1
    If datArmyRS.Recordset.EOF Then
        datArmyRS.Caption = "EOF"
    Else
        datArmyRS.Caption = "Record " & iCurrentRecord & " of " & iRecordCount
    End If
Else
    iRecordCount = datNavyRS.Recordset.RecordCount
    iCurrentRecord = datNavyRS.Recordset.AbsolutePosition + 1
    If datNavyRS.Recordset.EOF Then
        datNavyRS.Caption = "EOF"
    Else
        datNavyRS.Caption = "Record " & iCurrentRecord & " of " & iRecordCount
    End If
End If

End Sub

Private Sub cmdGoMain_Click()

On Error GoTo HandlecmdGoMainError

MIObj.Do ("Close All Interactive")

frmSelect.Hide
frmMain.Show
frmMain.cmdClear = True
frmMain.Refresh
cmdGoMain_Exit:
Exit Sub

```

```

HandleCmdGoMainError:
    MsgBox Err.Description
    Select Case Err.Number
        Case 91 'object or variable not found
            Resume
        Case Else
            MsgBox Err.Description, vbOKOnly + vbExclamation, "Unexpected Error"
    End Select
End Sub

Private Sub cmdRSSearch_Click()

    Call DisplayPanelMsg("Searching for RS")

    ' Determine service to search for
    Let strSearchFor2 = LCase(InputBox("Enter Service to search (Army, Navy)"))
    If strSearchFor2 <> "army" And strSearchFor2 <> "navy" Then
        Call DisplayMsg("Invalid service selection.", vbInformation, vbOKOnly, "Input Message")
        Exit Sub
    End If
    ' Search for the RS ID specified by the user
    Let strSearchFor = LCase(InputBox("Enter RS ID of the Recruiting Station ID: "))

    If Len(strSearchFor) > 0 Then
        If strSearchFor2 = "army" Then
            pctRSArmy.Visible = True
            pctArmyBN.Visible = True
            pctRSNavy.Visible = False
            pctNavyDistrict.Visible = False
            datArmyRS.Recordset.FindFirst "Ar_Rs = " & strSearchFor & ""
            If datArmyRS.Recordset.NoMatch Then
                MsgBox "No match found", vbExclamation, "Find an RS"
            End If
        ElseIf strSearchFor2 = "navy" Then
            pctRSNavy.Visible = True
            pctNavyDistrict.Visible = True
            pctArmyBN.Visible = False
            pctRSArmy.Visible = False
            datNavyRS.Recordset.FindFirst "Nv_Rs = " & strSearchFor & ""
            If datNavyRS.Recordset.NoMatch Then
                MsgBox "No match found", vbExclamation, "Find an RS"
            End If
        End If
    Else
        Call DisplayMsg("Must enter an R.S. ID.", vbInformation, vbOKOnly, "Input Message")
    End If

End Sub

Private Sub pctMapInfo_Click()

End Sub

Private Sub RS_PanelBar_PanelClick(ByVal Panel As Panel)

End Sub

```

```

Private Sub mnuFileExit_Click()

On Error GoTo HandleFileExitError

If cmdDoneGAMS.Visible = True Then
    strMsg = "You must first select DONE."
    Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "User Message")
    Exit Sub
Else

    'Close MapInfo files before killing. Otherwise error #75 results
    MIObj.Do "close all interactive"
    Kill (cMapPath & "Map*.*")      'All MapInfo files

    'close out objects and release all memory
    Set MIObj = Nothing
    Set datOptimize.Recordset = Nothing
    Set datTempQuery.Recordset = Nothing
    Set datTempQuery2.Recordset = Nothing
    Set datTempQuery2.Recordset = Nothing
    Set datValidateZip.Recordset = Nothing

    End
End If

mnuFileExit_Exit:
    Exit Sub

HandleFileExitError:
    Select Case Err.Number
        Case 75
            Resume Next
        Case Else
            MsgBox Err.Description, vbOKOnly + vbExclamation, "Unexpected Error"
            Resume Next
    End Select
End Sub

Private Sub mnuHelpAbout_Click()
    frmAbout.Show
End Sub

Private Sub mnuHelpRSLES_Click()

Dim OS As OSVERSIONINFO

strMsg = "Is your operating system Windows 95/98?"
iResponse = MsgBox(strMsg, vbYesNo + vbQuestion, "Input Message")
If iResponse = vbYes Then    'Let user input target production
    Select Case OS.dwPlatformId
        Case VER_PLATFORM_WIN32_WINDOWS
            Call Shell("c:\windows\hh.exe c:\Rsles\help\RslesHelp.chm", vbNormalFocus)
        Case VER_PLATFORM_WIN32s
            Call Shell("c:\windows\hh.exe c:\Rsles\help\RslesHelp.chm", vbNormalFocus)
        Case VER_PLATFORM_WIN32_NT
            Call Shell("c:\winnt\hh.exe c:\Rsles\help\RslesHelp.chm", vbNormalFocus)
    End Select
End If

```

```

'End Select
dTaskID = Shell("hh.exe c:\Program Files\Rsles\help\RslesHelp.chm", vbNormalFocus)
'Call Shell("c:\windows\hh.exe c:\Rsles\help\RslesHelp.chm", vbNormalFocus)
'Else
' Call Shell("c:\winnt\hh.exe c:\Rsles\help\RslesHelp.chm", vbNormalFocus)
'End If

End Sub

Private Sub mnuPrintStationtxt_Click()
Dim OS As OSVERSIONINFO

Call DisplayPanelMsg("Print Optimizer Output File")

On Error GoTo PrintStationtxtFile_errorhandler

strMsg = "Ensure the printer is on. Select FILE > PRINT from the Notepad menu."
Call DisplayMsg(strMsg, vbExclamation, vbOKOnly, "User Message")
dTaskID = Shell("notepad.exe c:\Program Files\Rsles\gams\stations.txt", vbNormalFocus)
strMsg = "Is your operating system Windows 95/98?"
iResponse = MsgBox(strMsg, vbYesNo + vbQuestion, "Input Message")
'If iResponse = vbYes Then 'Let user input target production
'Shell to notepad to let the user print the file
' Call Shell("c:\windows\notepad.exe c:\Program Files\Rsles\Gams\stations.txt", vbNormalFocus)
'Else
' Call Shell("c:\winnt\notepad.exe c:\Program Files\Rsles\Gams\stations.txt", vbNormalFocus)
'End If

'returns taskID
hprog = Shell("c:\Program Files\Rsles\notepad.exe c:\Rsles\gams\stations.txt", vbNormalFocus)

'Select Case OS.dwPlatformId
' Case VER_PLATFORM_WIN32_WINDOWS
' Call Shell("c:\windows\notepad.exe c:\Rsles\Gams\stations.txt", vbNormalFocus)
' Case VER_PLATFORM_WIN32s
' Call Shell("c:\windows\notepad.exe c:\Rsles\Gams\stations.txt", vbNormalFocus)
' Case VER_PLATFORM_WIN32_NT
' Call Shell("c:\winnt\notepad.exe c:\Rsles\Gams\stations.txt", vbNormalFocus)
'End Select

Do While IsActive(hprog)
DoEvents
Loop
'Get Process handle
hProc = OpenProcess(PROCESS_ALL_ACCESS, False, hprog)
'wait until the process terminates
If hProc <> 0 Then
RetVal = WaitForSingleObject(hProc, INFINITE)
CloseHandle hProc
End If

mnuPrintStationtxt_Exit:
Exit Sub

PrintStationtxtFile_errorhandler:

```

```

Select Case Err.Number
Case Else
    MsgBox "Error Number:" + Str(Err.Number) + _
        "; Description: " + Err.Description + _
        ". ", vbInformation, _
        "The Friendly Error Handler"
    Resume Next
End Select

End Sub

Private Sub mnuViewInstructions_Click()

strMsg = "Select 'HELP > RSLES HELP MENU' from the menu list." & vbCrLf & _
    "    View the instructions for 'Running RSLES'"
Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "User Message")

End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
    Dim cSelect As Double

    Select Case Button.Key
    Case "selector"
        MIObj.RunMenuCommand M_TOOLS_SELECTOR
        cSelect = MIObj.eval("MapperInfo(" & iMapInfoWinID & "," & MAPPER_INFO_ZOOM & ")")
        StatusBar.Panels(2).Text = "Zoom: " & cSelect & " Miles"
        StatusBar.Panels(3).Text = "MapInfo Tool: Selector"
    Case "grabber"
        MIObj.RunMenuCommand M_TOOLS_RECENTER
        StatusBar.Panels(3).Text = "MapInfo Tool: Grabber"

    Case "expand"
        MIObj.RunMenuCommand M_TOOLS_EXPAND
        cSelect = MIObj.eval("MapperInfo(" & iMapInfoWinID & "," & MAPPER_INFO_ZOOM & ")")
        StatusBar.Panels(2).Text = "Zoom: " & cSelect & " Miles"
        StatusBar.Panels(3).Text = "MapInfo Tool: Zoom-In"

    Case "shrink"
        MIObj.RunMenuCommand M_TOOLS_SHRINK
        cSelect = MIObj.eval("MapperInfo(" & iMapInfoWinID & "," & MAPPER_INFO_ZOOM & ")")
        StatusBar.Panels(2).Text = "Zoom: " & cSelect & " Miles"
        StatusBar.Panels(3).Text = "MapInfo Tool: Zoom-Out"

    Case "layer control"
        StatusBar.Panels(3).Text = "MapInfo Tool: Layer Control"
        MIObj.RunMenuCommand M_MAP_LAYER_CONTROL

    Case "info tool"
        MIObj.RunMenuCommand M_TOOLS_PNT_QUERY
        StatusBar.Panels(3).Text = "Custom Tool: RS_ID Info"
        ""Set the info window's parent to be the picture's hWnd
        MIObj.Do "Set Window " & WIN_INFO & " parent " & pctDisplay.hWnd

    Case "reset"
        MIObj.Do "Set Map Window " & iMapInfoWinID & " Zoom entire Layer 1"

```

```

        StatusBar.Panels(3).Text = "Reset Map Display"

    Case "Printer"
        StatusBar.Panels(3).Text = "Print Map"
        'Run Print Menu Command (From MIPro's File menu)
        MIObj.RunMenuCommand M_FILE_PRINT
    End Select
End Sub

Private Function cmdDoneSetupGAMSfiles()

    'Set screen view. Make Open,Close RS tab visible
    OpenCloseTabs.Visible = False
    cmdDoneGAMS.Visible = False
    cmdDoneSetup.Visible = False
    pctMeasure.Visible = True
    cmdResetMetrics.Visible = True
    cmdGoMain.Enabled = True
    cmdDisplayGAMS.Enabled = True
    cmdSetupOptimizer.Enabled = True
    cmdSetupOptimizer.Caption = "Setup Optimizer"
    frmLegend.Visible = True
    Call DisplayPanelMsg("Select 'Run' GAMS")

    'Reset ValidateZipCounter to 1
    ValidateZipCount = 1

    'Loop to clear control arrays for opening and closing
    For iindex = 1 To 10
        txtAOpenCloseRS(iindex) = ""
        txtAopenRS(iindex) = ""
        txtAcloseRS(iindex) = ""
        txtNopenRS(iindex) = ""
        txtNcloseRS(iindex) = ""
        txtNOpenCloseRS(iindex) = ""
    Next iindex

End Function

Private Function GetNewDatabaseName() As String
    'Allow user to browse for the database

    Dim stMsg As String

    stMsg = "Database file not found." & vbCrLf & vbCrLf & _
        "Do you want to locate the file?"
    iResponse = MsgBox(stMsg, vbYesNo + vbQuestion, "File or Path not Found")
    If iResponse = vbNo Then
        GetNewDatabaseName = "" 'Set return filename to empty string
    Else
        'Allow browse for file name
        With dlgOpen
            If Service = "Army" Then
                .FileName = datArmyRS.DatabaseName
            Else
                .FileName = datNavyRS.DatabaseName
            End If
        End With
    End If

```

```

.Filter = "Database files (*.mdb)|*.mdb|All files (*.*)|*.*"
On Error Resume Next      'Trap for the Cancel button
.ShowOpen                'Display Open dialog box
If Err.Number = cdlCancel Then 'Cancel button pressed
    GetNewDatabaseName = ""    'Set return filename to empty string
Else
    GetNewDatabaseName = .FileName 'Set return filename to selected file
End If
End With
End If

End Function

Public Function ValidateZipCode(ByVal cRSstatus As String, cZip As String, choice As String) As Boolean

'Develop Criteria for Search
ValidZipCode = True

If ValidateZipCount = 1 Then
    'Open Validation dynaset, find zip
    If cRSstatus = "ArmyRSClose" Or cRSstatus = "NavyRSClose" Then
        'Create dynaset that includes only zips with stations
        If Service = "Army" And frmMain.optDistrict.Value = True Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE astatzip = 1 and ar_battn = " & choice
        & " "
        ElseIf Service = "Army" And frmMain.optCoZone.Value = True Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE astatzip = 1 and ar_co = " & choice &
        " "
        ElseIf frmMain.optNavy.Value = True And frmMain.optDistrict.Value = True Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE nstatzip = 1 and nv_nrd = " & choice
        & " "
        ElseIf frmMain.optNavy.Value = True And frmMain.optCoZone.Value = True Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE nstatzip = 1 and nv_nrz = " & choice
        & " "
        ElseIf frmMain.optJoint.Value = True Then
            If cRSstatus = "ArmyRSClose" Then
                strSQL = "SELECT zip_code FROM Rslesdb WHERE astatzip = 1 and metro = " & choice
            & " "
            ElseIf cRSstatus = "NavyRSClose" Then
                strSQL = "SELECT zip_code FROM Rslesdb WHERE nstatzip = 1 and metro = " & choice
            & " "
        End If
    End If

    ElseIf cRSstatus = "ArmyRSOpen" Or cRSstatus = "NavyRSOpen" Then
        'Create dynaset that includes only zips w/no stations
        If Service = "Army" And frmMain.optDistrict.Value = True Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE astatzip = 0 and ar_battn = " & choice
        & " "
        ElseIf Service = "Army" And frmMain.optCoZone.Value = True Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE astatzip = 0 and ar_co = " & choice &
        " "
        ElseIf frmMain.optNavy.Value = True And frmMain.optDistrict.Value = True Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE nstatzip = 0 and nv_nrd = " & choice
        & " "
    End If
End If

```

```

ElseIf frmMain.optNavy.Value = True And frmMain.optCoZone.Value = True Then
    strSQL = "SELECT zip_code FROM Rslesdb WHERE nstatzip = 0 and nv_nrz = " & choice
& " "
ElseIf frmMain.optJoint.Value = True Then
    If cRSstatus = "ArmyRSOpen" Then
        strSQL = "SELECT zip_code FROM Rslesdb WHERE astatzip = 0 and metro = " & choice
& " "
    ElseIf cRSstatus = "NavyRSOpen" Then
        strSQL = "SELECT zip_code FROM Rslesdb WHERE nstatzip = 0 and metro = " & choice
& " "
    End If
End If
ElseIf cRSstatus = "ArmyRSOpenClose" Or cRSstatus = "NavyRSOpenClose" Then
    'Create dynaset that includes all zips in selected unit
    If Service = "Army" And frmMain.optDistrict.Value = True Then
        strSQL = "SELECT zip_code FROM Rslesdb WHERE ar_battn = " & choice & " "
    ElseIf Service = "Army" And frmMain.optCoZone.Value = True Then
        strSQL = "SELECT zip_code FROM Rslesdb WHERE ar_co = " & choice & " "
    ElseIf frmMain.optNavy.Value = True And frmMain.optDistrict.Value = True Then
        strSQL = "SELECT zip_code FROM Rslesdb WHERE nv_nrd = " & choice & " "
    ElseIf frmMain.optNavy.Value = True And frmMain.optCoZone.Value = True Then
        strSQL = "SELECT zip_code FROM Rslesdb WHERE nv_nrz = " & choice & " "
    ElseIf frmMain.optJoint.Value = True Then
        If cRSstatus = "ArmyRSOpenClose" Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE metro = " & choice & " "
        ElseIf cRSstatus = "NavyRSOpenClose" Then
            strSQL = "SELECT zip_code FROM Rslesdb WHERE metro = " & choice & " "
        End If
    End If
End If
datValidateZip.RecordSource = strSQL
datValidateZip.Refresh
ValidateZipCount = ValidateZipCount + 1
Else
End If

datValidateZip.Recordset.FindFirst "zip_code = " & cZip & ""
If datValidateZip.Recordset.NoMatch = True Then
    ValidZipCode = False
Else
    ValidZipCode = True
End If

End Function

Public Function QueryConvert(ByVal MapDynaset As Recordset, QFileName As String)
'Function to create files to be passed into MapInfo for Geocoding
Dim S$
Dim GeoFlag As Boolean
Dim i As Integer
Dim fld As Field

Open cMapPath & QFileName & ".tab" For Output As #1
Print #1, "!table"
Print #1, "!version 300"
Print #1, "!charset WindowsLatin1"

```

```

Print #1, ""
Print #1, "Definition Table"
Print #1, " File "" & QFileName & ".zap""
Print #1, " Type ASCII Delimiter 44 Titles Charset ""WindowsLatin1""
Print #1, " Fields " & MapDynaset.Fields.Count
For i = 0 To MapDynaset.Fields.Count - 1
    S$ = MapDynaset(i).Name
    Select Case S$
        Case "zip_code"
            Print #1, " " & S$ & " Char (5) ;"
            GeoFlag = True
        Case "Ar_Rs"
            Print #1, " " & S$ & " Char (4) ;"
        Case "Nv_Rs"
            Print #1, " " & S$ & " Char (6) ;"
        Case "pop17", "Tot17"
            Print #1, " " & S$ & " Integer (5) ;"
        Case "gsma_army", "gsma_dod"
            Print #1, " " & S$ & " Float (7.2) ;"
        Case "gsma_navy"
            Print #1, " " & S$ & " Float (7.2) ;"
        Case "areaname"
            Print #1, " " & S$ & " Char (66) ;"
        Case "TotHS", "NoHs"
            Print #1, " " & S$ & " Integer (7) ;"
        Case "MktShare"
            Print #1, " " & S$ & " Float (0.2) ;"
        Case Else
            Print #1, " " & S$ & " Integer ;"
    End Select

Next i
Close #1

Open cMapPath & QFileName & ".zap" For Output As #1
S$ = ""
For Each Fld In MapDynaset.Fields
    S$ = S$ & Chr$(34) & Fld.Name & Chr$(34) & Chr$(44)
Next Fld
S$ = Left(S$, Len(S$) - 1)
Print #1, S$
Debug.Print S$

While MapDynaset.EOF = False
    S$ = ""
    For Each Fld In MapDynaset.Fields
        If Fld.Type = dbText Then
            S$ = S$ & Chr$(34) & vFieldVal(Fld) & Chr$(34) & Chr$(44)
        Else
            S$ = S$ & vFieldVal(Fld) & Chr$(44)
        End If
    Next Fld
    S$ = Left(S$, Len(S$) - 1)
    Print #1, S$
    Debug.Print S$
    MapDynaset.MoveNext

```

```

Wend
Close #1

End Function
Private Function vFieldVal(fval As Variant) As Variant
    If IsNull(fval) Then
        vFieldVal = ""
    Else
        vFieldVal = CStr(fval)
    End If
End Function
Public Function FindArraySize(RSArray As Object, inMsg As String, inRSstatus As String, unit As String) As Integer
    'RSArray is a Control Array, inMsg is a user message, inRSstatus is the status of the RS
    'i.e. "ArmyRSOpen", unit is UnitSelected i.e. "Ar_battn"

    For iindex = 1 To 10
        If Trim(RSArray(iindex).Text) = "" And iindex = 1 Then
            iResponse = MsgBox(inMsg, vbYesNo + vbQuestion, "Input Message")
            If iResponse = vbNo Then
                'exit for loop and check to see if there are stations to fix
                Exit For
            Else
                ExitModule = True
                Exit Function
            End If
        Else
            If Trim(RSArray(iindex).Text) = "" And RSCount > 0 Then
                Exit For
            Else
                If Len(Trim(RSArray(iindex).Text)) <> 5 Then
                    Call DisplayMsg("A zip code must be 5 digits.", vbCrLf & _
                        vbInformation, vbOKOnly, "Input Message")
                    ExitModule = True
                    Exit Function
                Else
                    Call ValidateZipCode(inRSstatus, Trim(RSArray(iindex).Text), unit)
                    If ValidZipCode = True Then
                        RSCount = RSCount + 1
                    Else
                        strMsg = " Zip Code " & RSArray(iindex) & " is not in " & _
                            & UnitSelected & " or..." & vbCrLf & _
                            "the same zip code was entered twice." & vbCrLf & _
                            " Enter the correct zip code."
                        Call DisplayMsg(strMsg, vbInformation, vbOKOnly, "Data Entry Error")
                        ExitModule = True
                        Exit Function
                    End If
                End If
            End If
        End If
    Next iindex
    FindArraySize = RSCount
    'reset ValidateZip counter to 1
    ValidateZipCount = 1
End Function

```

Public Function SortCSTfile(ByVal iFilename As String)

* This function reads the records from gamsin.cst into parallel arrays. Each
 * array is then sorted. If there is a duplicate zip, the values from the latter
 * zip are used because that means they are zips entered by the user. If any
 * of the status fields contain a "9", they do NOT overwrite the status of the
 * previous record.
 *

Dim headerLine As String
 Dim iCount, icol, passNum, i As Integer
 Dim InData As Single
 Dim iZip() As Single, iCoststa() As Single, iCostusa() As Single, iCostusn() As Single
 Dim iCostrec() As Single, iCostj2() As Single

Open iFilename For Input As #2

'determine the number of lines in the file
 Line Input #2, headerLine ' skip header line
 iCount = 0
 Do While Not (EOF(2))
 For icol = 1 To 6
 Input #2, InData 'read the 6 fields from gams.cst file
 Next icol
 ' increment record counter by 1
 iCount = iCount + 1

Loop
 Close #2
 ReDim iZip(1 To iCount) As Single
 ReDim iCoststa(1 To iCount) As Single
 ReDim iCostusa(1 To iCount) As Single
 ReDim iCostusn(1 To iCount) As Single
 ReDim iCostrec(1 To iCount) As Single
 ReDim iCostj2(1 To iCount) As Single

Open iFilename For Input As #2

'populate the array
 Line Input #2, headerLine ' skip header line
 For i = 1 To iCount
 Input #2, iZip(i), iCoststa(i), iCostusa(i), iCostusn(i), iCostrec(i), iCostj2(i)
 Next i

Close #2

'sort zips in ascending order

For passNum = 1 To (iCount - 1)
 For i = 1 To (iCount - passNum)
 If iZip(i) > iZip(i + 1) Then 'switch them
 Call SwapNum(iZip(i), iZip(i + 1)) 'zip code
 Call SwapNum(iCoststa(i), iCoststa(i + 1)) 'Coststa
 Call SwapNum(iCostusa(i), iCostusa(i + 1)) 'Costusa
 Call SwapNum(iCostusn(i), iCostusn(i + 1)) 'Costusn
 Call SwapNum(iCostrec(i), iCostrec(i + 1)) 'Costrec
 Call SwapNum(iCostj2(i), iCostj2(i + 1)) 'costj2
 End If
 Next i

```

Next passNum

Open iFilename For Output As #2
'print header line
Print #2, Tab(9); "coststa"; Tab(18); "costusa"; Tab(27); "costusn"; _
    Tab(37); "costrec"; Tab(47); "costj2"
For i = 1 To iCount
    If i = iCount Then
        GoTo PrintLastRecord
    End If

    If iZip(i) = iZip(i + 1) Then 'print data from (i + 1)
        If Len(CStr(iZip(i + 1))) = 3 Then
            Print #2, "00" & CStr(iZip(i + 1));
        ElseIf Len(CStr(iZip(i + 1))) = 4 Then
            Print #2, "0" & CStr(iZip(i + 1));
        ElseIf Len(CStr(iZip(i + 1))) = 5 Then
            Print #2, CStr(iZip(i + 1)); 'Print zip, stay on the same line
        End If
        Print #2, Tab(8); iCoststa(i + 1); _
            Tab(17); iCostusa(i + 1); _
            Tab(26); iCostusn(i + 1); _
            Tab(36); iCostrec(i + 1); _
            Tab(46); iCostj2(i + 1)

        i = i + 1 'increment i by 1
    Else
PrintLastRecord:
        If Len(CStr(iZip(i))) = 3 Then
            Print #2, "00" & CStr(iZip(i));
        ElseIf Len(CStr(iZip(i))) = 4 Then
            Print #2, "0" & CStr(iZip(i));
        ElseIf Len(CStr(iZip(i))) = 5 Then
            Print #2, CStr(iZip(i)); 'Print zip, stay on the same line
        End If
        Print #2, Tab(8); iCoststa(i); _
            Tab(17); iCostusa(i); _
            Tab(26); iCostusn(i); _
            Tab(36); iCostrec(i); _
            Tab(46); iCostj2(i)
    End If

    If i = iCount Then
        Exit For
    End If

Next i
Close #2

End Function
Public Function SortSTDfile(ByVal iFilename As String)
*****
'* This function reads the records from gamsin.std into parallel arrays. Each
'* array is then sorted. If there is a duplicate zip, the values from the latter
'* zip are used because that means they are zips entered by the user. If any

```

```

** of the status fields contain a "9", they do NOT overwrite the status of the
** previous record.
**

```

```

*****

```

```

Dim headerLine As String
Dim iCount, icol, passNum, i As Integer
Dim InData As Single
Dim iZip() As Single, iLong() As Single, iLat() As Single, iArec() As Single
Dim iNrec() As Single, iLpop() As Single, iAstatus() As Single, iNstatus() As Single

```

```

Open iFilename For Input As #1

```

```

'determine the number of lines in the file
Line Input #1, headerLine ' skip header line
iCount = 0
Do While Not (EOF(1))
    For icol = 1 To 8
        Input #1, InData 'read the 11 fields from gams output file
    Next icol
    ' increment record counter by 1
    iCount = iCount + 1
Loop

```

```

Close #1

```

```

ReDim iZip(1 To iCount) As Single
ReDim iLong(1 To iCount) As Single
ReDim iLat(1 To iCount) As Single
ReDim iArec(1 To iCount) As Single
ReDim iNrec(1 To iCount) As Single
ReDim iLpop(1 To iCount) As Single
ReDim iAstatus(1 To iCount) As Single
ReDim iNstatus(1 To iCount) As Single

```

```

Open iFilename For Input As #1

```

```

'populate the array
Line Input #1, headerLine ' skip header line
For i = 1 To iCount
    Input #1, iZip(i), iLong(i), iLat(i), iArec(i), iNrec(i), iLpop(i), iAstatus(i), iNstatus(i)
Next i

```

```

Close #1

```

```

'sort zips in ascending order

```

```

For passNum = 1 To (iCount - 1)
    For i = 1 To (iCount - passNum)
        If iZip(i) > iZip(i + 1) Then 'switch them
            Call SwapNum(iZip(i), iZip(i + 1)) 'zip code
            Call SwapNum(iLong(i), iLong(i + 1)) 'lLong
            Call SwapNum(iLat(i), iLat(i + 1)) 'lLat
            Call SwapNum(iArec(i), iArec(i + 1)) 'Arec
            Call SwapNum(iNrec(i), iNrec(i + 1)) 'Nrec
            Call SwapNum(iLpop(i), iLpop(i + 1)) 'Lpop
            Call SwapNum(iAstatus(i), iAstatus(i + 1)) 'Astatus
            Call SwapNum(iNstatus(i), iNstatus(i + 1)) 'Nstatus
        End If
    Next i
Next passNum

```

```

Open iFilename For Output As #1
'print header line
Print #1, Tab(9); "lLong"; Tab(18); "lLat"; Tab(27); "aRec"; Tab(32); "nRec"; _
    Tab(39); "lPop"; Tab(47); "aStatus"; Tab(55); "nStatus"
For i = 1 To iCount
    If i = iCount Then
        GoTo PrintLastRecord
    End If

    If iZip(i) = iZip(i + 1) Then 'print data from (i + 1)
        If Len(CStr(iZip(i + 1))) = 3 Then
            Print #1, "00" & CStr(iZip(i + 1));
        ElseIf Len(CStr(iZip(i + 1))) = 4 Then
            Print #1, "0" & CStr(iZip(i + 1));
        ElseIf Len(CStr(iZip(i + 1))) = 5 Then
            Print #1, CStr(iZip(i + 1)); 'Print zip, stay on the same line
        End If
        Print #1, Tab(9); Format(Val(iLong(i + 1)), "###.000"); _
            Tab(18); Format(Val(iLat(i + 1)), "##.000"); _
            Tab(26); iArec(i + 1); _
            Tab(31); iNrec(i + 1); _
            Tab(38); iLpop(i + 1);
        If iAstatus(i + 1) = 9 Then
            Print #1, Tab(46); iAstatus(i); 'then use original value
        Else
            Print #1, Tab(46); iAstatus(i + 1); 'then use newest value
        End If
        If iNstatus(i + 1) = 9 Then
            Print #1, Tab(54); iNstatus(i)
        Else
            Print #1, Tab(54); iNstatus(i + 1)
        End If
        i = i + 1 'increment i by 1
    Else
        PrintLastRecord:
        If Len(CStr(iZip(i))) = 3 Then
            Print #1, "00" & CStr(iZip(i));
        ElseIf Len(CStr(iZip(i))) = 4 Then
            Print #1, "0" & CStr(iZip(i));
        ElseIf Len(CStr(iZip(i))) = 5 Then
            Print #1, CStr(iZip(i)); 'Print zip, stay on the same line
        End If
        Print #1, Tab(9); Format(Val(iLong(i)), "###.000"); _
            Tab(18); Format(Val(iLat(i)), "##.000"); _
            Tab(26); iArec(i); _
            Tab(31); iNrec(i); _
            Tab(38); iLpop(i);

        If iAstatus(i) = 9 Then 'check if first record has a 9
            Print #1, Tab(46); " 2"; 'was a zero 'if yes, overwrite
        Else
            Print #1, Tab(46); iAstatus(i);
        End If
        If iNstatus(i) = 9 Then
            Print #1, Tab(54); " 2" 'was a zero Changed by Prof Gue

```

```

Else
    Print #1, Tab(54); iNstatus(i)
End If
End If

If i = iCount Then
    Exit For
End If

Next i
Close #1

End Function

Private Sub SwapNum(a As Single, b As Single)
    Dim temp As Single
    Rem Interchange values of a and b
    Let temp = a
    Let a = b
    Let b = temp
End Sub

Public Function WriteCSTfile(ByVal inarray As Object, RSCount As Integer)
*****
'*
'* This function writes the cost data associated with the zip codes the user
'* entered into each control array (i.e. Stations to Open, Close, Open or Close
'* Each zip code with cost data are appended to the gamsin.cst file already
'* created.
'*
*****

On Error GoTo WriteCSTfile_errorhandler

If RSCount <> 0 Then

    Open cGamsPath & "gamsin.cst" For Append As #5
    For iindex = 1 To RSCount
        'Do query for each zip
        'Write Zip plus cost data
        strSQL = "SELECT zip_code,coststa, costusa, costusn,costrec,costj2 FROM Rslesdb "
        strSQL2 = "WHERE zip_code = " & inarray(iindex).Text & "" 'previously was Trim(...) wo/the 5,
        datOptimize.RecordSource = strSQL & strSQL2
        datOptimize.Refresh
        Print #5, datOptimize.Recordset!Zip_Code; _
            Tab(8); datOptimize.Recordset!coststa; _
            Tab(17); datOptimize.Recordset!costusa; _
            Tab(26); datOptimize.Recordset!costusn; _
            Tab(36); datOptimize.Recordset!costrec; _
            Tab(46); datOptimize.Recordset!costj2
    Next iindex
    Close #5
    Set datOptimize.Recordset = Nothing
Else
    Exit Function
End If

WriteCSTfile_Exit:

```

Exit Function

WriteCSTfile_errorhandler:

```
Select Case Err.Number
  Case 3021 'record not found
    Resume Next
  Case 94 'invalid use of null
    Call DisplayPanelMsg(RecordsetFailMSG)
    Resume Next
  Case 3004, 3024, 3044 'database, file, or path not found
    MsgBox Err.Description
    Resume Next
  Case 524
    Resume Next
  Case Else
    MsgBox "Error Number:" + Str(Err.Number) + _
      "; Description: " + Err.Description + _
      ". ", vbInformation, _
      "The Friendly Error Handler"
    Resume Next
End Select
```

End Function

Public Function WriteSTDfile(ByVal inarray As Object, RSCount As Integer, inRSstatus As Integer, Svc As String)

!*

!* This function writes the attributes associated with the zip codes the user
!* entered into each control array (i.e. Stations to Open, Close, Open or Close
!* Each zip code with attributes are appended to the gamsin.std file already
!* created.
!*

On Error GoTo WriteSTDfile_errorhandler

If RSCount <> 0 Then

```
Open cGamsPath & "gamsin.sti" For Append As #5
Open cGamsPath & "zzz.sti" For Append As #6
Open cGamsPath & "gamsin.std" For Append As #7
For iindex = 1 To RSCount
  Print #5, inarray(iindex).Text
  Print #6, inarray(iindex).Text; Tab(8); inarray(iindex).Text
  'Do query for each zip to open
  'Write fields for each query after the zip followed by status (0,1, or 2)
  strSQL = "SELECT zip_code,astatzip, nstatzip, Llong,Llat,Opra_Ar, Oprn_Nv,Pop17 FROM
Rslesdb "
  strSQL2 = "WHERE zip_code = " & inarray(iindex).Text & ""
  datOptimize.RecordSource = strSQL & strSQL2
  datOptimize.Refresh
  Print #7, datOptimize.Recordset!Zip_Code; _
    Tab(9); Format(Val(datOptimize.Recordset!Llong), "###.000"); _
    Tab(18); Format(Val(datOptimize.Recordset!Llat), "##.000"); _
```

```

Tab(26); Val(datOptimize.Recordset!opra_ar); _
Tab(31); Val(datOptimize.Recordset!Oprn_nv); _
Tab(38); Val(datOptimize.Recordset!pop17); _

If frmMain.optJoint = True Then
    'If joint, then print correct status
    If Svc = "A" Then
        Print #7, Tab(46); inRSstatus;
        Print #7, Tab(54); "9"
    ElseIf Svc = "N" Then
        Print #7, Tab(46); "9";
        Print #7, Tab(54); inRSstatus
    End If
Else
    If Service = "Army" Then
        Print #7, Tab(46); inRSstatus;
    Else
        'print default status
        Print #7, Tab(46); "9";
    End If
    If frmMain.optNavy = True Then
        Print #7, Tab(54); inRSstatus
    Else
        'print default status
        Print #7, Tab(54); "9"
    End If
End If

Next iindex
Close #5, #6, #7
Set datOptimize.Recordset = Nothing
Else
    Exit Function
End If

WriteSTDfile_Exit:
    Exit Function

WriteSTDfile_errorhandler:

Select Case Err.Number
    Case 3021 'record not found
        Resume Next
    Case 94 'invalid use of null
        Call DisplayPanelMsg(RecordsetFailMSG)
        Resume Next
    Case 3004, 3024, 3044 'database, file, or path not found
        MsgBox Err.Description
        Resume Next
    Case 524
        Resume Next
    Case Else
        MsgBox "Error Number:" + Str(Err.Number) + _
            "; Description: " + Err.Description + _
            ". ", vbInformation, _
            "The Friendly Error Handler"

```

```
        Resume Next
    End Select
```

```
End Function
```

```
Private Sub Form_Terminate()
```

```
'close out objects and release all memory
Set MIObj = Nothing
Set datOptimize.Recordset = Nothing
Set datTempQuery.Recordset = Nothing
Set datTempQuery2.Recordset = Nothing
```

```
End Sub
```

Module 3. SUBMAIN PROCEDURE

Purpose: Submain Procedure immediately executes upon RSLES application loading in memory. It is a standard Visual Basic initialization procedure. The procedure loads a splash screen while the Initial U/I form is loading and initializes the OLE connection with MapInfo components.

Source Type: Visual Basic for Applications

Source File: SubMain.bas

Code Listing:

```
Sub Main()  
    ' Show the splash screen while frmMain is loading.  
    Screen.MousePointer = vbHourglass  
    frmSplash1.Show  
  
    'Add your startup procedures here. ...  
    'Create the instance of the MapInfo object  
    Set MIObj = CreateObject("MapInfo.Application")  
  
    ' Show the main form and unload the splash screen.  
    frmMain.Show  
    Unload frmSplash1  
  
    'Set it so we should redraw the map  
    g_bRefreshMap = True  
  
End Sub
```

THIS PAGE LEFT INTENTIONALLY BLANK

Module 4. RSLES PUBLIC DECLARATIONS

Purpose: Initializes all public variables and constants.

Source Type: Visual Basic for Applications

Source File: PublicDeclarations.bas

Code Listing:

Option Explicit

```
' Public RSLES Global variables
'=====
Public MIObj As Object
Public MapDynaset As Recordset
Public iMapInfoWinID As Long 'Stores the handle to the Map
Public hprog, hProc, RetVal As Long 'used in shell functions
Public QFileName As String 'File to be passed into MapInfo containing User
                             ' selected View (Dist,Bn,Co,Zone)
Public strSQL, strSQL2, strSQL3 As String
Public strSQL4, strSQL5, strSQL6 As String
Public vBookMark As Variant
Public UnitSelected As String 'unit selected by user on main form
Public RSstatus As String 'used in Function ValidateZip (open,close,free)
Public selection As String
Public Service As String 'Name of Service (Army,Navy)
Public iResponse As Integer 'variable to see if user selects Yes/NO
Public unit, RSname, RSname2, cMapFileName As String
Public RSCount As Integer 'used in SetupGams & FindArraySize
Public iCount As Integer 'counter for # of lines of input data in arrays
Public InData As Single 'holds array values
Public iindex As Integer 'used as array subscript
Public strMsg, strSearchFor, strSearchFor2 As String
Public RunOptCount As Integer 'stores the # of optimizer runs
Public ValidateZipCount As Integer 'stores the # of calls to ValidateZip Function
Public ValidZipCode, ExitModule As Boolean
Public ComboBoxFilled As Boolean
Public g_bRefreshMap As Boolean

' RSLES File Paths
'=====
Public Const cPath As String = _
"c:\Rsles\"
Public Const cMapPath As String = _
"c:\Rsles\MapInfoTables\"
Public Const cDataPath As String = _
"c:\Rsles\Databasefiles\"
Public Const cGamsPath As String = _
"c:\Rsles\gams\"
```

' RSLES Status Bar Messages

'=====

Public Const SQLStatusMSG = "Calculating Measures for Facility : "

Public Const ClearButtonMSG = "Clears all User Entries"

Public Const ExitButtonMSG = "Ends Program"

' Help Panel/Status Display Messages

'=====

Public Const BuildGamsFilesMSG = "Creating files for GAMS..."

Public Const LaunchMapinfoMSG = "Loading Mapinfo and RS Databases"

Public Const ThematicMapMSG = "Creating a thematic map...."

Public Const RecordsetFailMSG = "Null values in data. Load FAILED."

Public Const PrinterReadyMSG = "Ensure Printer is Ready...."

' Rsles WinAPI32 Constants

'=====

Public Const SW_Normal = 3

Public Const SW_Minimum = 2

Public Const PROCESS_ALL_ACCESS = 0

' Rsles Error Handler Codes (Not used in this version of RSLES)

'=====

Public Const ERR_FileNotFound = 53

Public Const ERR_LockTableFailure = 3211

Public Const ERR_InvalidUseOfNull = 94

Public Const ERR_NoCurrentRecord = 3021

Public Const ERR_ObjectUnloaded = 364

Public Const ERR_NotValidPath = 3044

Public Const ERR_DatabaseAlreadyOpen = 3356

Public Const DefaultErrorValue = -999

' Public Functions used in RSLES

'=====

Public Declare Function sndPlaySound Lib "winmm.dll" Alias _
"sndPlaySoundA" (ByVal lpszSoundName As String, ByVal uFlags _
As Long) As Long

Module 5. MAPBASIC PUBLIC DECLARATIONS

Purpose: A library module that initializes all public variables and constants for communicating with the integrated mapping component.

Source Type: Visual Basic for Applications

Source File: MapBasic.bas

Code Listing:

' MapInfo version 4.0 - System defines

' This file contains defines useful when programming in the MapBasic language. There are three versions of this file:

' MAPBASIC.DEF - MapBasic syntax
' MAPBASIC.BAS - Visual Basic syntax
' MAPBASIC.H - C/C++ syntax

' The defines in this file are organized into the following sections:

' General Purpose defines:
' Macros, Logical constants, Angle conversion, Colors
' ButtonPadInfo() defines
' ColumnInfo() defines
' CommandInfo() defines
' FileAttr() defines
' IntersectNodes() parameters
' LayerInfo() defines
' MapperInfo() defines
' MenuItemInfoByID() and MenuItemInfoByHandler() defines
' ObjectGeography() defines
' ObjectInfo() defines
' SearchInfo() defines
' SelectionInfo() defines
' Server statement and function defines
' StringCompare() return values
' StyleAttr() defines
' SystemInfo() defines
' TableInfo() defines
' WindowInfo() defines
' Abbreviated list of error codes
' Backward Compatibility defines

' This file is converted into MAPBASIC.H by doing the following:

' - concatenate MAPBASIC.DEF and MENU.DEF into MAPBASIC.H
' - search & replace "" at beginning of a line with "/"
' - search & replace "Define" at beginning of a line with "#define"
' - delete the following sections:
' * General Purpose defines: Macros, Logical Constants, Angle Conversions
' * Abbreviated list of error codes
' * Backward Compatibility defines
' * Menu constants whose names have changed
' * Obsolete menu items

```
' This file is converted into MAPBASIC.BAS by doing the following:
' - concatenate MAPBASIC.DEF and MENU.DEF into MAPBASIC.BAS
' - search & replace "Define <name>" with "Global Const <name> ="
'   e.g. "<Define {[!-z]+} +{[!-z]}" with "Global Const \0 = \1" using Brief
' - delete the following sections:
'   * General Purpose defines: Macros, Logical Constants, Angle Conversions
'   * Abbreviated list of error codes
'   * Backward Compatibility defines
'   * Menu constants whose names have changed
'   * Obsolete menu items
```

```
=====
' General Purpose defines
=====
```

```
-----
' Colors
-----
```

```
Global Const BLACK = 0
Global Const WHITE = 16777215
Global Const RED = 16711680
Global Const GREEN = 65280
Global Const BLUE = 255
Global Const CYAN = 65535
Global Const MAGENTA = 16711935
Global Const YELLOW = 16776960
```

```
=====
' ButtonPadInfo() defines
=====
```

```
Global Const BTNPAD_INFO_FLOATING = 1
Global Const BTNPAD_INFO_WIDTH = 2
Global Const BTNPAD_INFO_NBTNS = 3
Global Const BTNPAD_INFO_X = 4
Global Const BTNPAD_INFO_Y = 5
Global Const BTNPAD_INFO_WINID = 6
```

```
=====
' ColumnInfo() defines
=====
```

```
Global Const COL_INFO_NAME = 1
Global Const COL_INFO_NUM = 2
Global Const COL_INFO_TYPE = 3
Global Const COL_INFO_WIDTH = 4
Global Const COL_INFO_DECPLACES = 5
Global Const COL_INFO_INDEXED = 6
Global Const COL_INFO_EDITABLE = 7
```

```
-----
' Column type defines, returned by ColumnInfo(<col_ref>, COL_INFO_TYPE)
-----
```

```
Global Const COL_TYPE_CHAR = 1
Global Const COL_TYPE_DECIMAL = 2
Global Const COL_TYPE_INTEGER = 3
Global Const COL_TYPE_SMALLINT = 4
Global Const COL_TYPE_DATE = 5
```

Global Const COL_TYPE_LOGICAL = 6
Global Const COL_TYPE_GRAPHIC = 7
Global Const COL_TYPE_FLOAT = 8

=====

' CommandInfo() defines

=====

Global Const CMD_INFO_X = 1
Global Const CMD_INFO_Y = 2
Global Const CMD_INFO_SHIFT = 3
Global Const CMD_INFO_CTRL = 4
Global Const CMD_INFO_X2 = 5
Global Const CMD_INFO_Y2 = 6
Global Const CMD_INFO_TOOLBTN = 7
Global Const CMD_INFO_MENUITEM = 8
Global Const CMD_INFO_WIN = 1
Global Const CMD_INFO_SELTYP = 1
Global Const CMD_INFO_ROWID = 2
Global Const CMD_INFO_INTERRUPT = 3
Global Const CMD_INFO_STATUS = 1
Global Const CMD_INFO_MSG = 1000
Global Const CMD_INFO_DLG_OK = 1
Global Const CMD_INFO_DLG_DBL = 1
Global Const CMD_INFO_FIND_RC = 3
Global Const CMD_INFO_FIND_ROWID = 4
Global Const CMD_INFO_XCMD = 1
Global Const CMD_INFO_CUSTOM_OBJ = 1
Global Const CMD_INFO_TASK_SWITCH = 1

=====

' Task Switch, returned by CommandInfo(CMD_INFO_TASK_SWITCH)

=====

Global Const SWITCHING_OUT_OF_MAPINFO = 0
Global Const SWITCHING_INTO_MAPINFO = 1

=====

' FileAttr() defines

=====

Global Const FILE_ATTR_MODE = 1
Global Const FILE_ATTR_FILESIZE = 2

=====

' File Access modes, returned by FileAttr(<file_id>, FILE_ATTR_MODE)

=====

Global Const MODE_INPUT = 0
Global Const MODE_OUTPUT = 1
Global Const MODE_APPEND = 2
Global Const MODE_RANDOM = 3
Global Const MODE_BINARY = 4

=====

' IntersectNodes(obj1, obj2, mode) parameters

=====

Global Const INCL_CROSSINGS = 1
Global Const INCL_COMMON = 6
Global Const INCL_ALL = 7

```
'=====
' LayerInfo() defines
'=====
```

```
Global Const LAYER_INFO_NAME = 1
Global Const LAYER_INFO_EDITABLE = 2
Global Const LAYER_INFO_SELECTABLE = 3
Global Const LAYER_INFO_ZOOM_LAYERED = 4
Global Const LAYER_INFO_ZOOM_MIN = 5
Global Const LAYER_INFO_ZOOM_MAX = 6
Global Const LAYER_INFO_COSMETIC = 7
Global Const LAYER_INFO_PATH = 8
Global Const LAYER_INFO_DISPLAY = 9
Global Const LAYER_INFO_OVR_LINE = 10
Global Const LAYER_INFO_OVR_PEN = 11
Global Const LAYER_INFO_OVR_BRUSH = 12
Global Const LAYER_INFO_OVR_SYMBOL = 13
Global Const LAYER_INFO_OVR_FONT = 14
Global Const LAYER_INFO_LBL_EXPR = 15
Global Const LAYER_INFO_LBL_LT = 16
Global Const LAYER_INFO_LBL_CURFONT = 17
Global Const LAYER_INFO_LBL_FONT = 18
Global Const LAYER_INFO_LBL_PARALLEL = 19
Global Const LAYER_INFO_LBL_POS = 20
Global Const LAYER_INFO_ARROWS = 21
Global Const LAYER_INFO_NODES = 22
Global Const LAYER_INFO_CENTROIDS = 23
Global Const LAYER_INFO_TYPE = 24
Global Const LAYER_INFO_LBL_VISIBILITY = 25
Global Const LAYER_INFO_LBL_ZOOM_MIN = 26
Global Const LAYER_INFO_LBL_ZOOM_MAX = 27
Global Const LAYER_INFO_LBL_AUTODISPLAY = 28
Global Const LAYER_INFO_LBL_OVERLAP = 29
Global Const LAYER_INFO_LBL_DUPLICATES = 30
Global Const LAYER_INFO_LBL_OFFSET = 31
Global Const LAYER_INFO_LBL_MAX = 32
```

```
'-----
' Display Modes, returned by LayerInfo() for LAYER_INFO_DISPLAY
'-----
```

```
Global Const LAYER_INFO_DISPLAY_OFF = 0
Global Const LAYER_INFO_DISPLAY_GRAPHIC = 1
Global Const LAYER_INFO_DISPLAY_GLOBAL = 2
Global Const LAYER_INFO_DISPLAY_VALUE = 3
```

```
'-----
' Label Linetypes, returned by LayerInfo() for LAYER_INFO_LBL_LT
'-----
```

```
Global Const LAYER_INFO_LBL_LT_NONE = 0
Global Const LAYER_INFO_LBL_LT_SIMPLE = 1
Global Const LAYER_INFO_LBL_LT_ARROW = 2
```

```
'-----
' Label Positions, returned by LayerInfo() for LAYER_INFO_LBL_POS
'-----
```

```
Global Const LAYER_INFO_LBL_POS_CC = 0
```

```

Global Const LAYER_INFO_LBL_POS_TL = 1
Global Const LAYER_INFO_LBL_POS_TC = 2
Global Const LAYER_INFO_LBL_POS_TR = 3
Global Const LAYER_INFO_LBL_POS_CL = 4
Global Const LAYER_INFO_LBL_POS_CR = 5
Global Const LAYER_INFO_LBL_POS_BL = 6
Global Const LAYER_INFO_LBL_POS_BC = 7
Global Const LAYER_INFO_LBL_POS_BR = 8

```

```

'-----
' Layer Types, returned by LayerInfo() for LAYER_INFO_TYPE
'-----

```

```

Global Const LAYER_INFO_TYPE_NORMAL = 0
Global Const LAYER_INFO_TYPE_COSMETIC = 1
Global Const LAYER_INFO_TYPE_IMAGE = 2
Global Const LAYER_INFO_TYPE_THEMATIC = 3

```

```

'-----
' Label visibility modes, returned by LayerInfo() for LAYER_INFO_LBL_VISIBILITY
'-----

```

```

Global Const LAYER_INFO_LBL_VIS_OFF = 1
Global Const LAYER_INFO_LBL_VIS_ZOOM = 2
Global Const LAYER_INFO_LBL_VIS_ON = 3

```

```

'=====
' MapperInfo() defines
'=====

```

```

Global Const MAPPER_INFO_ZOOM = 1
Global Const MAPPER_INFO_SCALE = 2
Global Const MAPPER_INFO_CENTERX = 3
Global Const MAPPER_INFO_CENTERY = 4
Global Const MAPPER_INFO_MINX = 5
Global Const MAPPER_INFO_MINY = 6
Global Const MAPPER_INFO_MAXX = 7
Global Const MAPPER_INFO_MAXY = 8
Global Const MAPPER_INFO_LAYERS = 9
Global Const MAPPER_INFO_EDIT_LAYER = 10
Global Const MAPPER_INFO_XYUNITS = 11
Global Const MAPPER_INFO_DISTUNITS = 12
Global Const MAPPER_INFO_AREAUNITS = 13
Global Const MAPPER_INFO_SCROLLBARS = 14
Global Const MAPPER_INFO_DISPLAY = 15
Global Const MAPPER_INFO_NUM_THEMATIC = 16
Global Const MAPPER_INFO_COORDSYS_CLAUSE = 17
Global Const MAPPER_INFO_COORDSYS_NAME = 18

```

```

'-----
' Display Modes, returned by MapperInfo() for MAPPER_INFO_DISPLAY
'-----

```

```

Global Const MAPPER_INFO_DISPLAY_SCALE = 0
Global Const MAPPER_INFO_DISPLAY_ZOOM = 1
Global Const MAPPER_INFO_DISPLAY_POSITION = 2

```

```

'=====
' MenuItemInfoByID() and MenuItemInfoByHandler() defines
'=====

```

```

Global Const MENUITEM_INFO_ENABLED = 1
Global Const MENUITEM_INFO_CHECKED = 2
Global Const MENUITEM_INFO_CHECKABLE = 3
Global Const MENUITEM_INFO_SHOWHIDEABLE = 4
Global Const MENUITEM_INFO_ACCELERATOR = 5
Global Const MENUITEM_INFO_TEXT = 6
Global Const MENUITEM_INFO_HELPMSG = 7
Global Const MENUITEM_INFO_HANDLER = 8
Global Const MENUITEM_INFO_ID = 9

```

```

=====
' ObjectGeography() defines
=====

```

```

Global Const OBJ_GEO_MINX = 1
Global Const OBJ_GEO_LINEBEGX = 1
Global Const OBJ_GEO_POINTX = 1
Global Const OBJ_GEO_MINY = 2
Global Const OBJ_GEO_LINEBEGY = 2
Global Const OBJ_GEO_POINTY = 2
Global Const OBJ_GEO_MAXX = 3
Global Const OBJ_GEO_LINEENDX = 3
Global Const OBJ_GEO_MAXY = 4
Global Const OBJ_GEO_LINEENDY = 4
Global Const OBJ_GEO_ARCBEGANGLE = 5
Global Const OBJ_GEO_TEXTLINEX = 5
Global Const OBJ_GEO_ROUNDRAADIUS = 5
Global Const OBJ_GEO_ARCENDANGLE = 6
Global Const OBJ_GEO_TEXTLINEY = 6
Global Const OBJ_GEO_TEXTANGLE = 7

```

```

=====
' ObjectInfo() defines
=====

```

```

Global Const OBJ_INFO_TYPE = 1
Global Const OBJ_INFO_PEN = 2
Global Const OBJ_INFO_SYMBOL = 2
Global Const OBJ_INFO_TEXTFONT = 2
Global Const OBJ_INFO_BRUSH = 3
Global Const OBJ_INFO_NPNTS = 20
Global Const OBJ_INFO_TEXTSTRING = 3
Global Const OBJ_INFO_SMOOTH = 4
Global Const OBJ_INFO_FRAMEWIN = 4
Global Const OBJ_INFO_NPOLYGONS = 21
Global Const OBJ_INFO_TEXTSPACING = 4
Global Const OBJ_INFO_TEXTJUSTIFY = 5
Global Const OBJ_INFO_FRAMETITLE = 6
Global Const OBJ_INFO_TEXTARROW = 6

```

```

-----
' Object types, returned by ObjectInfo(<obj>, OBJ_INFO_TYPE)
-----

```

```

Global Const OBJ_TYPE_ARC = 1
Global Const OBJ_TYPE_ELLIPSE = 2
Global Const OBJ_TYPE_LINE = 3
Global Const OBJ_TYPE_PLINE = 4
Global Const OBJ_TYPE_POINT = 5

```

```

Global Const OBJ_TYPE_FRAME = 6
Global Const OBJ_TYPE_REGION = 7
Global Const OBJ_TYPE_RECT = 8
Global Const OBJ_TYPE_ROUNDRECT = 9
Global Const OBJ_TYPE_TEXT = 10

```

```

' =====
' SearchInfo() defines
' =====

```

```

Global Const SEARCH_INFO_TABLE = 1
Global Const SEARCH_INFO_ROW = 2

```

```

' =====
' SelectionInfo() defines
' =====

```

```

Global Const SEL_INFO_TABLENAME = 1
Global Const SEL_INFO_SELNAME = 2
Global Const SEL_INFO_NROWS = 3

```

```

' =====
' Server statement and function defines
' =====

```

```

' =====
' Return Codes
' =====

```

```

Global Const SRV_SUCCESS = 0
Global Const SRV_SUCCESS_WITH_INFO = 1
Global Const SRV_ERROR = -1
Global Const SRV_INVALID_HANDLE = -2
Global Const SRV_NEED_DATA = 99
Global Const SRV_NO_MORE_DATA = 100

```

```

' =====
' Special values for the status associated with a fetched value
' =====

```

```

Global Const SRV_NULL_DATA = -1
Global Const SRV_TRUNCATED_DATA = -2

```

```

' =====
' Server_ColumnInfo() Attr defines
' =====

```

```

Global Const SRV_COL_INFO_NAME = 1
Global Const SRV_COL_INFO_TYPE = 2
Global Const SRV_COL_INFO_WIDTH = 3
Global Const SRV_COL_INFO_PRECISION = 4
Global Const SRV_COL_INFO_SCALE = 5
Global Const SRV_COL_INFO_VALUE = 6
Global Const SRV_COL_INFO_STATUS = 7
Global Const SRV_COL_INFO_ALIAS = 8

```

```

' =====
' Column types, returned by Server_ColumnInfo(<stmt>,<colno>,SRV_COL_INFO_TYPE)
' =====

```

```

Global Const SRV_COL_TYPE_NONE = 0
Global Const SRV_COL_TYPE_CHAR = 1
Global Const SRV_COL_TYPE_DECIMAL = 2

```

```

Global Const SRV_COL_TYPE_INTEGER = 3
Global Const SRV_COL_TYPE_SMALLINT = 4
Global Const SRV_COL_TYPE_DATE = 5
Global Const SRV_COL_TYPE_LOGICAL = 6
Global Const SRV_COL_TYPE_FLOAT = 8
Global Const SRV_COL_TYPE_FIXED_LEN_STRING = 16
Global Const SRV_COL_TYPE_BIN_STRING = 17

```

```

'-----
' Server_DriverInfo() Attr defines
'-----

```

```

Global Const SRV_DRV_INFO_NAME = 1
Global Const SRV_DRV_INFO_NAME_LIST = 2
Global Const SRV_DRV_DATA_SOURCE = 3

```

```

'-----
' Fetch Directions used by Server_Fetch()
'-----

```

```

Global Const SRV_FETCH_NEXT = -1
Global Const SRV_FETCH_PREV = -2
Global Const SRV_FETCH_FIRST = -3
Global Const SRV_FETCH_LAST = -4

```

```

'=====
' StringCompare(<str_1>, <str_2>) return values
'=====

```

```

Global Const STR_LT = -1
Global Const STR_GT = 1
Global Const STR_EQ = 0

```

```

'=====
' StyleAttr() defines
'=====

```

```

Global Const PEN_WIDTH = 1
Global Const PEN_PATTERN = 2
Global Const PEN_COLOR = 4
Global Const BRUSH_PATTERN = 1
Global Const BRUSH_FORECOLOR = 2
Global Const BRUSH_BACKCOLOR = 3
Global Const FONT_NAME = 1
Global Const FONT_STYLE = 2
Global Const FONT_POINTSIZE = 3
Global Const FONT_FORECOLOR = 4
Global Const FONT_BACKCOLOR = 5
Global Const SYMBOL_CODE = 1
Global Const SYMBOL_COLOR = 2
Global Const SYMBOL_POINTSIZE = 3
Global Const SYMBOL_ANGLE = 4
Global Const SYMBOL_FONT_NAME = 5
Global Const SYMBOL_FONT_STYLE = 6
Global Const SYMBOL_KIND = 7
Global Const SYMBOL_CUSTOM_NAME = 8
Global Const SYMBOL_CUSTOM_STYLE = 9

```

```

'-----
' Symbol kinds returned by StyleAttr(<symbol>, SYMBOL_KIND)

```

```
'-----  
Global Const SYMBOL_KIND_VECTOR = 1  
Global Const SYMBOL_KIND_FONT = 2  
Global Const SYMBOL_KIND_CUSTOM = 3
```

```
'-----  
' SystemInfo() defines  
'-----
```

```
Global Const SYS_INFO_PLATFORM = 1  
Global Const SYS_INFO_APPVERSION = 2  
Global Const SYS_INFO_MIVERSION = 3  
Global Const SYS_INFO_RUNTIME = 4  
Global Const SYS_INFO_CHARSET = 5  
Global Const SYS_INFO_COPYPROTECTED = 6  
Global Const SYS_INFO_APPLICATIONWND = 7  
Global Const SYS_INFO_DDESTATUS = 8  
Global Const SYS_INFO_MAPINFOWND = 9  
Global Const SYS_INFO_NUMBER_FORMAT = 10  
Global Const SYS_INFO_DATE_FORMAT = 11  
Global Const SYS_INFO_DIG_INSTALLED = 12  
Global Const SYS_INFO_DIG_MODE = 13  
Global Const SYS_INFO_MIPLATFORM = 14  
Global Const SYS_INFO_MDICLIENTWND = 15  
Global Const SYS_INFO_PRODUCTLEVEL = 16
```

```
'-----  
' Platform, returned by SystemInfo(SYS_INFO_PLATFORM)  
'-----
```

```
Global Const PLATFORM_SPECIAL = 0  
Global Const PLATFORM_WIN = 1  
Global Const PLATFORM_MAC = 2  
Global Const PLATFORM_MOTIF = 3  
Global Const PLATFORM_X11 = 4  
Global Const PLATFORM_XOL = 5
```

```
'-----  
' Version, returned by SystemInfo(SYS_INFO_MIPLATFORM)  
'-----
```

```
Global Const MIPLATFORM_SPECIAL = 0  
Global Const MIPLATFORM_WIN16 = 1  
Global Const MIPLATFORM_WIN32 = 2  
Global Const MIPLATFORM_POWERMAC = 3  
Global Const MIPLATFORM_MAC68K = 4  
Global Const MIPLATFORM_HP = 5  
Global Const MIPLATFORM_SUN = 6
```

```
'-----  
' TableInfo() defines  
'-----
```

```
Global Const TAB_INFO_NAME = 1  
Global Const TAB_INFO_NUM = 2  
Global Const TAB_INFO_TYPE = 3  
Global Const TAB_INFO_NCOLS = 4  
Global Const TAB_INFO_MAPPABLE = 5  
Global Const TAB_INFO_READONLY = 6  
Global Const TAB_INFO_TEMP = 7
```

```

Global Const TAB_INFO_NROWS = 8
Global Const TAB_INFO_EDITED = 9
Global Const TAB_INFO_FASTEDIT = 10
Global Const TAB_INFO_UNDO = 11
Global Const TAB_INFO_MAPPABLE_TABLE = 12
Global Const TAB_INFO_USERMAP = 13
Global Const TAB_INFO_USERBROWSE = 14
Global Const TAB_INFO_USERCLOSE = 15
Global Const TAB_INFO_USEREDITABLE = 16
Global Const TAB_INFO_USERREMOVEMAP = 17
Global Const TAB_INFO_USERDISPLAYMAP = 18
Global Const TAB_INFO_TABFILE = 19
Global Const TAB_INFO_MINX = 20
Global Const TAB_INFO_MINY = 21
Global Const TAB_INFO_MAXX = 22
Global Const TAB_INFO_MAXY = 23
Global Const TAB_INFO_SEAMLESS = 24
Global Const TAB_INFO_COORDSYS_MINX = 25
Global Const TAB_INFO_COORDSYS_MINY = 26
Global Const TAB_INFO_COORDSYS_MAXX = 27
Global Const TAB_INFO_COORDSYS_MAXY = 28
Global Const TAB_INFO_COORDSYS_CLAUSE = 29
Global Const TAB_INFO_COORDSYS_NAME = 30
Global Const TAB_INFO_NREFS = 31

```

```

'-----
' Table type defines, returned by TableInfo(<tab_ref>, TAB_INFO_TYPE)
'-----

```

```

Global Const TAB_TYPE_BASE = 1
Global Const TAB_TYPE_RESULT = 2
Global Const TAB_TYPE_VIEW = 3
Global Const TAB_TYPE_IMAGE = 4
Global Const TAB_TYPE_LINKED = 5

```

```

'=====
' WindowInfo() defines
'=====

```

```

Global Const WIN_INFO_NAME = 1
Global Const WIN_INFO_TYPE = 3
Global Const WIN_INFO_WIDTH = 4
Global Const WIN_INFO_HEIGHT = 5
Global Const WIN_INFO_X = 6
Global Const WIN_INFO_Y = 7
Global Const WIN_INFO_TOPMOST = 8
Global Const WIN_INFO_STATE = 9
Global Const WIN_INFO_TABLE = 10
Global Const WIN_INFO_LEGENDS_MAP = 10
Global Const WIN_INFO_OPEN = 11
Global Const WIN_INFO_WND = 12
Global Const WIN_INFO_WINDOWID = 13
Global Const WIN_INFO_WORKSPACE = 14
Global Const WIN_INFO_CLONEWINDOW = 15
Global Const WIN_INFO_SYSMENUCLOSE = 16
Global Const WIN_INFO_AUTOSCROLL = 17

```

```

'-----

```

' Window types, returned by WindowInfo(<win_id>, WIN_INFO_TYPE)

Global Const WIN_MAPPER = 1
Global Const WIN_BROWSER = 2
Global Const WIN_LAYOUT = 3
Global Const WIN_GRAPH = 4
Global Const WIN_BUTTONPAD = 19
Global Const WIN_HELP = 1001
Global Const WIN_MAPBASIC = 1002
Global Const WIN_MESSAGE = 1003
Global Const WIN_RULER = 1007
Global Const WIN_INFO = 1008
Global Const WIN_LEGEND = 1009
Global Const WIN_STATISTICS = 1010
Global Const WIN_MAPINFO = 1011

' Version 2 window types no longer used in version 3 or version 4

Global Const WIN_TOOLPICKER = 1004
Global Const WIN_PENPICKER = 1005
Global Const WIN_SYMBOLPICKER = 1006

' Window states, returned by WindowInfo(<win_id>, WIN_INFO_STATE)

Global Const WIN_STATE_NORMAL = 0
Global Const WIN_STATE_MINIMIZED = 1
Global Const WIN_STATE_MAXIMIZED = 2

=====

' Set Next Document Style defines

=====

Global Const WIN_STYLE_STANDARD = 0
Global Const WIN_STYLE_CHILD = 1
Global Const WIN_STYLE_POPUP_FULLCAPTION = 2
Global Const WIN_STYLE_POPUP = 3

=====

' end of MAPBASIC.DEF

=====

=====

' MapInfo version 4.0 - Menu Item Definitions

' This file contains defines useful when programming in the MapBasic
' language. The definitions in this file describe the standard MapInfo
' functionality available via the "Run Menu Command" MapBasic statement.

' The defines in this file are organized to match the sequence of
' declarations in the MAPINFOW.MNU file, which in turn reflects the
' organization of the MapInfo menus and buttonpads.
=====

' File & Send Mail menus

Global Const M_FILE_NEW = 101
 Global Const M_FILE_OPEN = 102
 Global Const M_FILE_OPEN_ODBC = 116
 Global Const M_FILE_ADD_WORKSPACE = 108
 Global Const M_FILE_CLOSE = 103
 Global Const M_FILE_CLOSE_ALL = 104
 Global Const M_FILE_SAVE = 105
 Global Const M_FILE_SAVE_COPY_AS = 106
 Global Const M_FILE_SAVE_WORKSPACE = 109
 Global Const M_FILE_SAVE_WINDOW_AS = 609
 Global Const M_FILE_REVERT = 107
 Global Const M_FILE_RUN = 110
 Global Const M_FILE_PAGE_SETUP = 111
 Global Const M_FILE_PRINT = 112
 Global Const M_FILE_EXIT = 113

 Global Const M_SENDMAIL_CURRENTWINDOW = 114
 Global Const M_SENDMAIL_WORKSPACE = 115

' Edit menu

Global Const M_EDIT_UNDO = 201
 Global Const M_EDIT_CUT = 202
 Global Const M_EDIT_COPY = 203
 Global Const M_EDIT_PASTE = 204
 Global Const M_EDIT_CLEAR = 205
 Global Const M_EDIT_CLEAROBJ = 206
 Global Const M_EDIT_RESHAPE = 1601
 Global Const M_EDIT_NEW_ROW = 702
 Global Const M_EDIT_GETINFO = 207

' Objects menu

Global Const M_OBJECTS_SET_TARGET = 1610
 Global Const M_OBJECTS_CLEAR_TARGET = 1611
 Global Const M_OBJECTS_COMBINE = 1605
 Global Const M_OBJECTS_SPLIT = 1612
 Global Const M_OBJECTS_ERASE = 1613
 Global Const M_OBJECTS_ERASE_OUT = 1614
 Global Const M_OBJECTS_OVERLAY = 1615
 Global Const M_OBJECTS_BUFFER = 1606
 Global Const M_OBJECTS_SMOOTH = 1602
 Global Const M_OBJECTS_UNSMOOTH = 1603
 Global Const M_OBJECTS_CVT_PGON = 1607
 Global Const M_OBJECTS_CVT_PLINE = 1604

' Query menu

Global Const M_ANALYZE_SELECT = 301
 Global Const M_ANALYZE_SQLQUERY = 302
 Global Const M_ANALYZE_SELECTALL = 303
 Global Const M_ANALYZE_UNSELECT = 304
 Global Const M_ANALYZE_FIND = 305

Global Const M_ANALYZE_FIND_SELECTION = 306
Global Const M_ANALYZE_CALC_STATISTICS = 309

' Table, Maintenance, and Raster menus

Global Const M_TABLE_UPDATE_COLUMN = 405
Global Const M_TABLE_APPEND = 411
Global Const M_TABLE_GEOCODE = 407
Global Const M_TABLE_CREATE_POINTS = 408
Global Const M_TABLE_MERGE_USING_COLUMN = 406
Global Const M_TABLE_IMPORT = 401
Global Const M_TABLE_EXPORT = 402

Global Const M_TABLE_MODIFY_STRUCTURE = 404
Global Const M_TABLE_DELETE = 409
Global Const M_TABLE_RENAME = 410
Global Const M_TABLE_PACK = 403
Global Const M_TABLE_MAKEMAPPABLE = 415
Global Const M_TABLE_UNLINK = 416
Global Const M_TABLE_REFRESH = 417

Global Const M_TABLE_RASTER_STYLE = 414
Global Const M_TABLE_RASTER_REG = 413
Global Const M_TOOLS_RASTER_REG = 1730

' Options menu

Global Const M_FORMAT_PICK_LINE = 501
Global Const M_FORMAT_PICK_FILL = 502
Global Const M_FORMAT_PICK_SYMBOL = 503
Global Const M_FORMAT_PICK_FONT = 504
Global Const M_WINDOW_BUTTONPAD = 605
Global Const M_WINDOW_LEGEND = 606
Global Const M_WINDOW_STATISTICS = 607
Global Const M_WINDOW_MAPBASIC = 608
Global Const M_WINDOW_STATUSBAR = 616
Global Const M_FORMAT_CUSTOM_COLORS = 617
Global Const M_EDIT_PREFERENCES = 208
Global Const M_EDIT_PREFERENCES_SYSTEM = 210
Global Const M_EDIT_PREFERENCES_FILE = 211
Global Const M_EDIT_PREFERENCES_MAP = 212
Global Const M_EDIT_PREFERENCES_COUNTRY = 213
Global Const M_EDIT_PREFERENCES_PATH = 214

' Window menu

Global Const M_WINDOW_BROWSE = 601
Global Const M_WINDOW_MAP = 602
Global Const M_WINDOW_GRAPH = 603
Global Const M_WINDOW_LAYOUT = 604
Global Const M_WINDOW_REDISTRIBUTE = 615
Global Const M_WINDOW_REDRAW = 610
Global Const M_WINDOW_TILE = 611

Global Const M_WINDOW_CASCADE = 612
Global Const M_WINDOW_ARRANGEICONS = 613
Global Const M_WINDOW_MORE = 614
Global Const M_WINDOW_FIRST = 620

'-----
' Note: the 2nd through 80th windows can be accessed as (M_WINDOW_FIRST+i-1)
'-----

'-----
' Help menu
'-----

Global Const M_HELP_CONTENTS = 1202
Global Const M_HELP_SEARCH = 1203
Global Const M_HELP_USE_HELP = 1204
Global Const M_HELP_TECHSUPPORT = 1208
Global Const M_HELP_CONNECT_MIFORUM = 1209
Global Const M_HELP_ABOUT = 1205

Global Const M_HELP_CONTEXTSENSITIVE = 1201
Global Const M_HELP_HELPMODE = 1206

'-----
' Browse menu
'-----

Global Const M_BROWSE_PICK_FIELDS = 704
Global Const M_BROWSE_OPTIONS = 703

'-----
' Map menu
'-----

Global Const M_MAP_LAYER_CONTROL = 801
Global Const M_MAP_THEMATIC = 307
Global Const M_MAP_MODIFY_THEMATIC = 308
Global Const M_MAP_CHANGE_VIEW = 805
Global Const M_MAP_CLONE_MAPPER = 811
Global Const M_MAP_PREVIOUS = 806
Global Const M_MAP_ENTIRE_LAYER = 807
Global Const M_MAP_CLEAR_CUSTOM_LABELS = 814
Global Const M_MAP_SAVE_COSMETIC = 809
Global Const M_MAP_CLEAR_COSMETIC = 810
Global Const M_MAP_SET_CLIP_REGION = 812
Global Const M_MAP_CLIP_REGION_ONOFF = 813
Global Const M_MAP_SETUPDIGITIZER = 803
Global Const M_MAP_OPTIONS = 802

'-----
' Layout menu
'-----

Global Const M_LAYOUT_CHANGE_VIEW = 902
Global Const M_LAYOUT_ACTUAL = 903
Global Const M_LAYOUT_ENTIRE = 904
Global Const M_LAYOUT_PREVIOUS = 905
Global Const M_LAYOUT_BRING2FRONT = 906
Global Const M_LAYOUT_SEND2BACK = 907
Global Const M_LAYOUT_ALIGN = 908
Global Const M_LAYOUT_DROPSHADOWS = 909

Global Const M_LAYOUT_DISPLAYOPTIONS = 901

' Graph menu

Global Const M_GRAPH_TYPE = 1001
Global Const M_GRAPH_LABEL_AXIS = 1002
Global Const M_GRAPH_VALUE_AXIS = 1003
Global Const M_GRAPH_SERIES = 1004

' MapBasic menu

Global Const M_MAPBASIC_CLEAR = 1101
Global Const M_MAPBASIC_SAVECONTENTS = 1102

' Redistrict menu

Global Const M_REDISTRICK_ASSIGN = 705
Global Const M_REDISTRICK_TARGET = 706
Global Const M_REDISTRICK_ADD = 707
Global Const M_REDISTRICK_DELETE = 708
Global Const M_REDISTRICK_OPTIONS = 709

' Main Buttonpad

Global Const M_TOOLS_SELECTOR = 1701
Global Const M_TOOLS_SEARCH_RECT = 1722
Global Const M_TOOLS_SEARCH_RADIUS = 1703
Global Const M_TOOLS_SEARCH_BOUNDARY = 1704
Global Const M_TOOLS_EXPAND = 1705
Global Const M_TOOLS_SHRINK = 1706
Global Const M_TOOLS_RECENTER = 1702
Global Const M_TOOLS_PNT_QUERY = 1707
Global Const M_TOOLS_LABELER = 1708
Global Const M_TOOLS_DRAGWINDOW = 1734
Global Const M_TOOLS_RULER = 1710

' Drawing Buttonpad

Global Const M_TOOLS_POINT = 1711
Global Const M_TOOLS_LINE = 1712
Global Const M_TOOLS_POLYLINE = 1713
Global Const M_TOOLS_ARC = 1716
Global Const M_TOOLS_POLYGON = 1714
Global Const M_TOOLS_ELLIPSE = 1715
Global Const M_TOOLS_RECTANGLE = 1717
Global Const M_TOOLS_ROUNDEDRECT = 1718
Global Const M_TOOLS_TEXT = 1709
Global Const M_TOOLS_FRAME = 1719
Global Const M_TOOLS_ADD_NODE = 1723

' Menu and ButtonPad items that do not appear in the standard menus

Global Const M_TOOLS_MAPBASIC = 1720

Global Const M_TOOLS_SEARCH_POLYGON = 1733

' end of MENU.DEF

'*-----*

' Consts for cursors built into MapBasic.

' Available to be used with the ButtonPads.

'*-----*

Global Const MI_CURSOR_ARROW = 0

Global Const MI_CURSOR_IBEAM = 1

Global Const MI_CURSOR_FINGER_LEFT = 128

Global Const MI_CURSOR_ZOOM_IN = 129

Global Const MI_CURSOR_ZOOM_OUT = 130

Global Const MI_CURSOR_DRAG_OBJ = 131

Global Const MI_CURSOR_GRABBER = 132

Global Const MI_CURSOR_CHANGE_WIDTH = 133

Global Const MI_CURSOR_FINGER_UP = 134

Global Const MI_CURSOR_IBEAM_CROSS = 135

Global Const MI_CURSOR_CROSSHAIR = 138

'*-----*

' Consts for different DrawModes for the custom tool.

'*-----*

Global Const DM_CUSTOM_CIRCLE = 30

Global Const DM_CUSTOM_ELLIPSE = 31

Global Const DM_CUSTOM_RECT = 32

Global Const DM_CUSTOM_LINE = 33

Global Const DM_CUSTOM_POINT = 34

Global Const DM_CUSTOM_POLYGON = 35

Global Const DM_CUSTOM_POLYLINE = 36

APPENDIX E. INSTALLATION AND SYSTEM NOTES

Recruit Station Location and Evaluation System (RSLES)

Installation and System Notes for Windows 95/98, Windows NT

1. SYSTEM REQUIREMENTS

To install the system you need approximately 76 MB of disk space (this does not include the GAMS, Access, or MapInfo applications). At least 32 MB of RAM is required to run the optimizer model, but 64 MB or more, is highly recommended. A math Coprocessor is required and it is recommended that a Pentium or higher Intel compatible chip be used.

2. INSTALLATION (all Windows commands are shown in bold italics)

STEP 1 – Install the RSLES system files

- a. Insert the RSLES CD into your computer's CD-ROM.
- b. Click on **START > RUN > BROWSE**. Locate and select the setup.exe file located on your CD-ROM. Click OK. This will install the RSLES program to the C:\Program Files\Rsles directory.
- c. The RSLES Setup Wizard will appear and will first copy some system files to your computer. When the wizard reappears, click **OK**.
- d. The RSLES Setup window will appear. The default installation directory will be c:\Program Files\Rsles **NOTE:** You must accept this default. RSLES will not run properly if installed to any other directory.
- e. Click the button to install RSLES. The Choose Program Group window will appear with a default group of RSLES. Click **Continue** to accept this default.

- f. The Setup Wizard will the install the required Data Access Components (DAO) followed by the RSLES system files. This process will take a few minutes.
NOTE: If a Version Conflict window appears, select Yes to keep the newer files.
- g. Click **OK** when the installation is completed.
- h. The following folders will be created in the c:\Program Files\Rsles directory:
- Rsles: Contains the executable file for RSLES (rslocation.exe)
 - Databasefiles: This folder contains the Rsles.mdb file. This is a Microsoft Access database.
 - Gams: This folder contains the files required to run the optimizer model in GAMS. The installed files are runit.bat (see step 4), armynavy2.gms, and armynavy2.inc. This folder will also contain the files necessary for the optimizer model (see the RSLES Help Menu under *Files Required by GAMS*) and the output from the optimizer model, station.txt and armynavy2.lst.
 - Help: This folder contains the files necessary for the RSLES Help Menu.
 - MapInfoTables: This folder contains the MapInfo tables required to run RSLES.
 - Source Code: This folder contains the Visual Basic files that are used by RSLES.

STEP 2 – Create a shortcut on your desktop (optional)

- a. Create a shortcut for RSLES on your desktop.
- b. Open Windows Explorer. Locate the Rslocation.exe file in the directory:
c:\Program Files\Rsles
- c. Right click on the file. Select 'Create Shortcut'
- d. Drag and drop the shortcut to the desktop.

STEP 3 – Add the GAMS directory to your path

(if this was not already done during GAMS installation)

a. This is required to access GAMS from any directory on your machine. RSLES requires that GAMS version 2.25.089 or later be installed to your C:\ drive (if you plan on using the optimizing portion of RSLES). If GAMS is already installed on the c:\ drive of your computer, double check to ensure that the GAMS directory is in your path by following steps c and d. If GAMS is not installed, install it to the C:\Gams directory following the instructions provided by the GAMS Development Corporation. (<http://www.gams.com>).

b. Install GAMS (with CPLEX solver) to the directory c:\Gams

c. For Windows 95: Edit the autoexec.bat file

- Add the GAMS directory to the path as follows:

path=c:\windows;c:\windows\command;c:\gams

d. For Windows NT:

- Open the System Properties under the Control Panel
- On the Environment tab click on the existing variable PATH
- In the Value box, add the GAMS directory to the pat as follows:

(%SystemRoot%\system32;%SystemRoot%;c:\gams

- Click set.

STEP 4 – Run RSLES

a. Click **Start > RSLES > RSLES DSS** The RSLES group should be located in the Program Group portion of the Windows Start menu.

LIST OF REFERENCES

1. Boehm, Barry W., "A Spiral Model of Software Development and Enhancement," *Computer*, May 1988.
2. Bohn, John; Schwartz, Edward; VanMeter, Richard, *Analysis Report—The Geographic Location of Recruiting Resources*. Commander, Naval Recruiting Command. June 1996.
3. Booch, Grady, *Object-Oriented Analysis and Design with Applications*, Addison-Wesley, Menlo Park, CA, 1997.
4. Brackett, Michael H., *The Data Warehouse Challenge*, John Wiley & Sons, Inc., New York, NY, 1996.
5. Chappell, David, *The Next Wave: Component Software Enters The Mainstream*, Chappell & Associates White Paper, April 1997, URL: <http://www.rational.com/support/techpapers/nextwave/nextwave.html>
6. Falk, Peter R., *Aries: An Architectural Implementation of a Multi-criterion Spatial Decision Support System (SDSS)*, Master's Thesis, Naval Postgraduate School, September, 1997.
7. Greenfield, Adam, "From RAD to Riches", *VARbusiness*, December 1, 1997.
8. Harris, John B., Eichorn, Frank L., Goodwin, David L., and Henson, Joel L., "Use of Rapid Application Development Techniques," *Logistics Management Institute*, September, 1997.
9. Litwin, Paul, Getz, K., Gilbert, M., and Reddick, G., *Access 97 Developer's Handbook*, SYBEX, Alameda, California, 1996.
10. Luqi, "Increasing the Practical Impact of Formal Methods for Computer-Aided Software Development: Specification-Based Software Architectures," Proceedings of the 1995 Monterey Workshop, September 1995.
11. Maner, Walter, "Rapid Application Development", URL: <http://web.cs.edu/maner/domains/RAD.html>, May 1997
12. Martin, Paul E., *A Multi-Service Location-Allocation Model For Military Recruiting*, Master's Thesis, Naval Postgraduate School, March 1999.
13. McManus, Jeffrey P., *Database Access with Visual Basic*, Sams Publishing, Indianapolis, Indiana, 1998.
14. MDA Computing, "Iterative", URL: <http://www.mdagroup.com/computing/iterativ.htm>
15. Mehay, Stephen. *OSD Recruiting Station Location Project*. Presentation. March 1998.
16. Mize, Rick. "Recruiters Lament: 'It's the Economy,'" *The Army Times*. Army Times Publishing Company, Springfield, VA, April 15, 1999.

17. Murphy, Mark A., *An Automated Spatial Decision Support System for the Relocation of Army Reserve Units*, Master's Thesis, Naval Postgraduate School, March 1997.
18. O'Brien, James A., *Management Information Systems: A Managerial End User Perspective*, Irwin, Homewood, Illinois, 1993.
19. Office of the Assistant Secretary of Defense (Force Management Policy), Report to the House Committee on National Security, Senate Committee on Armed Services, House and Senate Committees on Appropriations; Study of a Joint Process for Determining the Location of Recruiting Stations, Second Interim Report. November 1996.
20. Waugh, Douglas, "*Prospectus on Software Architecture*," Proceedings of the 1995 Monterey Workshop, September 1995.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
8725 John J. Kingman Rd., STE 0944
Ft. Belvoir, VA, 22060-6218

2. Dudley Knox Library..... 2
Naval Postgraduate School
411 Dyer Rd.
Monterey, CA 93943-5101

3. Director, Training and Education.....1
MCCDC, Code C46
1019 Elliot Rd.
Quantico, VA 22134-5027

4. Director, Marine Corps Research Center.....2
MCCDC, Code C40RC
300 Russell Road
Quantico, VA 22134-5107

5. Director, Studies and Analysis Division.....1
MCCDC, Code C45
300 Russell Road
Quantico, VA 22134-5130

6. Marine Corps Representative 1
Naval Postgraduate School
Code 037, Bldg 330
555 Dyer Road
Monterey, CA 93940

7. Marine Corps Tactical Systems Support Activity..... 1
Technical Advisory Branch
Attn: Maj J.S. Cummiskey
Box 555171
Camp Pendleton, CA 92055-5080

8. Headquarters, U.S. Army Recruiting Command..... 2
ATTN: PAE-MKT
1307 3d Avenue
Ft. Knox, KY 40121-2726

9. Commander, Navy Recruiting Command.....2
ATTN: John Noble
5720 Integrity Drive BLDG 784
Millington, TN 38954

10. Professor Daniel Dolk, Code SM/Dk 2
 Department of Systems Management
 Naval Postgraduate School
 555 Dyer Rd Bldg 330
 Monterey, California 93943-5000
11. Professor Kevin Gue, Code SM/GK 2
 Department of Systems Management
 Naval Postgraduate School
 555 Dyer Rd Bldg 330
 Monterey, California 93943-5000
12. Professor Stepen Mehay, Code SM/MP 1
 Department of Systems Management
 Naval Postgraduate School
 555 Dyer Rd, Bldg 330
 Monterey, California 93943-5000
13. Major Dale E. Houck 2
 223 Colmar Rd.
 Seaside, California 93955
14. Major Mark V. Shigley 2
 209 Naples Rd.
 Seaside, California 93955